ALIBABA CLOUD

阿里云

函数计算

函数计算公共云合集

文档版本：20220712

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.新功能发布记录 09
2.客户案例 22
2.1. 世纪联华 22
2.2. 捷旅假期 22
2.3. 石墨文档 24
2.4. 新浪微博 24
2.5. 闲鱼 25
2.6. 语雀 26
2.7. 清林云 29
2.8. 高德 30
2.9. 拉勾教育 32
3.权限管理 34
3.1. 配置权限助手 34
3.2. 授予函数计算访问其他云服务的权限 38
3.3. 授予事件源访问函数计算的权限 40
3.4. 通过阿里云账号给RAM用户授权 41
3.5. 通过RAM角色实现跨云账号授权 43
3.6. 权限策略及示例 45
3.7. 权限管理FAQ 52
3.7.1. 如何让其他用户为我的函数授予访问其他资源的权限？ 52
4.服务管理 53
4.1. 管理服务 53
4.2. 管理版本 55
4.3. 管理别名 56
4.4. 灰度发布示例 58
4.5. 配置网络 61

4.6. 配置固定公网IP地址 64
4.7. 配置NAS文件系统 67
4.8. 配置日志 70
4.9. 管理标签 73
4.10. 服务管理FAQ 77
4.10.1. 什么是VPC? 77
4.10.2. 如何用函数访问私有VPC资源? 77
4.10.3. 函数可以访问多少个VPC? 77
4.10.4. 我在VPC环境里如何访问经典网络中的资源? 77
4.10.5. 在VPC中的函数如何访问公网? 77
4.10.6. 遇到vSwitch is in unsupported zone的错误怎么办? 78
4.10.7. 开启VPC功能对可用性的影响 78
5.函数管理 80
5.1. 管理函数 80
5.2. 为函数安装第三方依赖 83
5.3. 层管理 84
5.3.1. 构建层 84
5.3.2. 使用层 88
5.4. 环境变量 91
5.5. 在函数中配置自定义DNS 94
5.6. 函数管理FAQ 96
5.6.1. 函数运行FAQ 96
5.6.1.1. 如何监控函数执行? 96
5.6.1.2. 可执行文件在本地测试正常，但在函数计算的运行环境中执行.. 96
5.6.1.3. 如果程序要产生很大一个文件，或者需要一个比较大的空间，.. 96
5.6.1.4. 在程序运行过程中，能从哪些接口获取到实际使用的内存? 97
5.6.1.5. 函数可以相互调用吗? 97
5.6.1.6. 函数如果出现死循环，怎么杀掉函数进程? 97

5.6.1.7. 如何避免循环触发？ 97
5.6.1.8. 我的执行环境是否是独立安全的？怎么保障？ 97
5.6.1.9. 临时磁盘空间什么时候被释放？ 97
5.6.1.10. 当前目录不可写怎么办？ 98
5.6.1.11. 我的函数会运行很久，超过函数最大Timeout时间怎么办？ 98
5.6.1.12. 我有10个函数，都用到了同一个公共库。我能否只上传一份... 98
5.6.1.13. 当我运行Python语言的函数时，遇到报错NoneType obje... 98
5.6.1.14. 函数返回后，执行环境也会随之释放吗？我能复用上一次调... 99
5.6.2. 函数调试FAQ 100
5.6.2.1. 如何快速定位线上问题？ 100
5.6.2.2. 为什么我的Node.js函数莫名其妙地超时了？ 100
5.6.2.3. 遇到The service or function doesn't belong to you的... 100
5.6.2.4. 遇到SDK.ServerUnreachable：SocketTimeoutExceptio... 100
5.6.2.5. 为什么通过浏览器访问HTTP Trigger对应的函数会强制下载... 100
5.6.2.6. 如何确定函数内部访问其他服务的网络连通性问题？ 100
5.6.2.7. 如何在Custom Runtime中引用层中的依赖？ 100
6.高级功能 102
6.1. 配置自定义域名 102
6.2. 操作审计 110
7.访问其他服务 111
7.1. 访问NAS示例 111
7.2. 访问数据库 112
7.3. 访问RDS MySQL示例 113
7.4. 访问MongoDB示例 115
7.5. 访问Redis示例 118
7.6. 访问RDS SQL Server示例 120
7.7. 访问RDS PostgreSQL数据库 122
7.8. 访问Lindorm数据库 123

7.9. 访问其他服务FAQ 126
7.9.1. 函数计算如何访问RDS? 126
7.9.2. 为什么我在函数里无法通过内网Endpoint访问OSS等云服务? 126
7.9.3. 在函数中如何访问其他服务? 126
8.Serverless Devs 127
8.1. 从Funcraft迁移到Serverless Devs 127
8.2. 什么是Serverless Devs 128
8.3. 安装Serverless Devs 129
8.4. 配置Serverless Devs 131
8.5. 使用Serverless Devs管理函数资源 133
8.6. Serverless Devs操作命令 139
8.7. FC-API组件 140
8.7.1. FC-API组件权限信息 140
8.7.2. 服务相关命令 143
8.7.3. 函数相关命令 145
8.7.4. 触发器相关命令 149
8.7.5. 版本相关命令 152
8.7.6. 别名相关命令 154
8.7.7. 自定义域名相关命令 156
8.7.8. 预留配置相关命令 158
8.7.9. 函数异步调用相关命令 160
8.8. 使用Serverless Devs迁移Web框架 161
9.常见问题 168
10.联系我们 171
11.安全白皮书 172
11.1. 概述 172
11.2. 控制面安全性 172
11.3. 数据面安全性 173

11.4. 责任共担 ---------- 175

1.新功能发布记录

本文介绍函数计算FC的产品功能和对应的文档动态。

2022年04月

功能名称	功能描述	发布时间	相关文档
HTTP函数支持异步调用	HTTP函数用于快速构建Web应用，HTTP触发器支持异步调用和异步任务。如果您的HTTP函数中存在耗时较长、资源消耗较大或容易出错的逻辑，您可以使用异步调用的方式，让您的程序响应更加迅速，更加可靠地面对突发流量。	2022-04-18	• HTTP触发器概述 • 配置HTTP触发器并使用HTTP触发 • 异步调用 • 异步任务
固定公网IP地址	函数计算访问数据库、微信小程序或其他第三方服务时，需通过配置白名单来访问，您可以使用固定公网IP地址完成白名单的验证。	2022-04-11	配置固定公网IP地址

2022年03月

功能名称	功能描述	发布时间	相关文档
闲置计费（Beta）	闲置计费功能是针对预留模式函数实例的功能。函数计算默认为预留模式的实例分配CPU，如果启用**闲置计费**功能，则只为活跃实例分配CPU，不为闲置实例分配CPU。闲置实例资源使用单价是活跃实例资源使用单价的20%，这将为您节省大量成本。	2022-03-31	• 弹性管理（含预留模式） • 计费概述
HTTP触发器域名更换（公测中）	HTTP触发器主域名更换为 `fcapp.run` 。函数计算支持为所有新创建的HTTP触发器创建 `fcapp.run` 域名的子域名。使用子域名访问函数可以避免在URL中填写服务名称、函数名称和版本别名信息等。	2022-03-21	• 配置HTTP触发器并使用HTTP触发 • 当我使用浏览器或cURL方式访问函数时出现404怎么办？ • 使用fcapp.run运行您的RESTful应用

2021年12月

功能名称	功能描述	发布时间	相关文档
实例命令行操作	实例命令行操作功能能够支持在实例的真实运行环境中执行指定命令，例如登录进入实例查看实例环境信息，或者使用profiling或coredump等工具收集上下文信息来优化性能等。	2021-12-21	函数实例命令行操作

功能名称	功能描述	发布时间	相关文档
权限助手	权限助手是一个生成RAM权限策略的工具，能够协助您对函数计算的权限进行可视化配置，并在函数计算控制台生成对应的RAM权限语句。然后，您可以在RAM控制台创建对应的自定义权限策略，将权限策略的策略内容修改为函数计算控制台生成的RAM权限语句，并为需要该项权限的RAM用户授予权限。	2021-12-20	• 配置权限助手 • 权限策略及示例

2021年10月

功能名称	功能描述	发布时间	相关文档
GPU实例（公测中）	基于Turing架构的GPU实例，主要适用于音视频、AI人工智能和图像处理等场景。在不同的场景中，将不同的业务负载下沉至GPU硬件加速，从而极大地提升了业务处理的效率。	2021-10-18	• 实例类型及使用模式 • 图像处理最佳实践 • 人工智能最佳实践 • 音视频处理最佳实践
EventBridge触发器	EventBridge触发器是指事件总线EventBridge作为阿里云官方事件源和自定义事件源的事件中心，支持阿里云官方事件源和自定义事件源作为触发源触发关联函数执行，实现事件总线EventBridge与函数计算的无缝集成。	2021-10-15	• EventBridge触发器简介 • RocketMQ触发器 • RabbitMQ触发器 • MNS队列触发器 • 官方事件源触发器

2021年08月

功能名称	功能描述	发布时间	相关文档

功能名称	功能描述	发布时间	相关文档

Serverless Devs	Serverless Devs是一个开源开放的Serverless开发者平台，通过Serverless Devs，您不仅可以可插拔式地使用Serverless的服务和框架，也可以参与组件和插件的开发，提高运维效率。同时，您也可以更简单、快速地开发、创建、测试和部署项目，实现项目全生命周期的管理。	2021-08-13	• 从Funcraft迁移到Serverless Devs
			• 什么是Serverless Devs
			• 安装Serverless Devs
			• 配置Serverless Devs
			• Serverless Devs操作命令
			• 使用Serverless Devs迁移Web框架
			• FC-API组件
			  ◦ FC-API组件权限信息
			  ◦ 服务相关命令
			  ◦ 函数相关命令
			  ◦ 触发器相关命令
			  ◦ 版本相关命令
			  ◦ 别名相关命令
			  ◦ 自定义域名相关命令
			  ◦ 预留配置相关命令
			  ◦ 函数异步调用相关命令

2021年06月

功能名称	功能描述	发布时间	相关文档
资源套餐包（预付费）	资源套餐包是一种资源计费方式，套餐内的资源可以在函数计算中优先抵扣资源使用量。购买资源套餐包您可以以更优惠的价格享受等量资源，从而减少支出。	2021-06-25	资源套餐包
有状态异步调用	有状态异步调用可以保存调用执行过程中的状态转换信息，提升调用的可靠性。除函数执行实例以外的硬件故障外，都将不会导致函数执行的中断或重复执行函数。 有状态异步调用适用于以下场景： • 函数执行时间长。 • 需要查看每次执行的结果。 • 在执行函数的过程中需要中断函数的执行。	2021-06-09	• 同步调用 • 功能概览 • GetStatefulAsyncInvocation • ListStatefulAsyncInvocations • StopStatefulAsyncInvocation

2021年05月

功能名称	功能描述	发布时间	相关文档

功能名称	功能描述	发布时间	相关文档
默认服务角色（AliyunFCDefaultRole）	在函数运行的过程中，函数计算需要访问其他云资源，例如将函数日志写入到您指定的日志服务内、拉取ACR镜像或打通VPC网络访问等。为了简化您的授权操作，函数计算为您提供了一个系统默认的服务角色即AliyunFCDefaultRole。该角色内包含了函数计算需要访问的部分云资源权限。	2021-06-02	• 开通服务 • 授予函数计算访问其他云服务的权限

2021年01月

功能名称	功能描述	发布时间	相关文档
层管理	函数计算新增层管理功能，您可以将函数依赖的公共库提炼到层，以减少部署、更新时的代码包体积，也可以将自定义的Runtime以层部署，在多个函数间共享。	2021-01-15	• 构建层 • 使用层 • PublishLayerVersion • DeleteLayerVersion • GetLayerVersion • ListLayerVersions • ListLayers

2020年12月

功能名称	功能描述	发布时间	相关文档
CICD部署	函数计算新增CICD部署功能，通过CICD部署功能，您无需管理服务器等基础设置，只需编写代码并上传。函数计算已为您准备好计算资源，并以弹性、可靠的方式运行您的代码。	2020-12-29	• 使用函数计算控制台部署函数 • 使用云效2020部署函数
计费粒度调整	按量模式下弹性实例的计费粒度由原来的百毫秒更改为毫秒。	2020-12-15	实例类型及使用模式

2020年11月

功能名称	功能描述	发布时间	相关文档
调用分析	调用分析功能是函数请求级别的执行状态汇总，开启调用分析功能后，系统会收集函数每次执行的指标信息，包括内存使用情况、函数执行时间、初始化时间、冷启动信息等性能指标，是否执行失败、错误详情等异常指标，链路追踪详情、是否采样等追踪指标，并将这些指标信息投递到您在日志配置时选择的日志仓库中。开启调用分析可以洞察每一次函数调用，帮助您掌握函数执行情况。	2020-11-30	• 调用分析简介 • 配置调用分析

功能名称	功能描述	发布时间	相关文档
镜像拉取加速	容器镜像相比于函数代码包更容易在不同技术栈之间移植和迁移，也具有围绕容器镜像的丰富且标准的工具链生态和最佳实践。然而容器镜像自带的操作系统、命令工具以及其他与应用本身不需要的文件不可避免地加大了需要下载和解压的数据量，增加了函数冷启动时间。对于较大的镜像，如解压前1 GB以上级别的容器镜会造成分钟级别的镜像拉取延迟，明显拖慢了函数实例的冷启动性能。因此函数计算针对容器镜像拉取做了一系列优化，在不同场景下可以获得2~5倍的加速提升，将分钟级的镜像拉取缩短至几秒。	2020-11-30	镜像启动加速（ACR个人版）
ARMS高级监控	函数计算无缝对接ARMS应用监控，您可以通过配置环境变量获得ARMS的APM应用监控功能，ARMS将对Java 8运行环境的应用进行无侵入零代码改动的高级监控，获得实例级别的可观测性，为您提供更丰富的指标，例如总请求量、响应时间及异常信息等。	2020-11-30	Java函数监控
预留模式实例的自动弹性伸缩	函数计算新增预留模式实例的自动弹性伸缩功能，您可以通过定时弹性伸缩和指标追踪弹性伸缩两种模式解决预留模式配置的固定预留值导致的预留函数实例利用不充分问题。	2020-11-30	预留模式实例的自动弹性伸缩
编程模式扩展	函数计算针对以下痛点发布了运行时扩展（runtime extensions）功能。该功能在现有的HTTP服务编程模型上扩展，在已有的HTTP服务器的模型中增加了PreFreeze和PreStop webhooks。扩展开发者实现HTTP handler，监听函数实例生命周期事件。编程模式扩展解决以下痛点： • 异步背景指标数据延迟或丢失。 • 同步发送指标增加延迟。 • 函数优雅下线。	2020-11-26	编程模式扩展
操作审计	函数计算新增了操作审计ActionTrail功能，操作审计ActionTrail提供了行为操作日志，可以对您的行为进行追踪、查看及分析。	2020-11-25	操作审计

2020年10月

功能名称	功能描述	发布时间	相关文档
Serverless Devs	Serverless Devs Tool是一款让Serverless开发者开发及运维效率增加的工具。通过该工具，您可以更简单、更快速的进行项目的开发、创建、测试和部署，实现项目全生命周期的管理。	2020-10-23	Serverless Devs Tool

2020年09月

功能名称	功能描述	发布时间	相关文档

功能名称	功能描述	发布时间	相关文档
函数异步调用配置	函数计算支持异步调用的相关配置，让您可以在函数执行成功或者失败后将执行结果发送给其他目标服务，您也可以设置异步调用的重试次数和消息存活时长。	2020-09-25	• 功能概览 • PutFunctionAsyncInvokeConfig • DeleteFunctionAsyncInvokeConfig • GetFunctionAsyncInvokeConfig • ListFunctionAsyncInvokeConfigs
自定义容器镜像	在云原生时代，容器镜像已经逐渐变成了软件部署和开发的标准工具。为了简化开发者体验、提升开发和交付效率，函数计算提供了Custom Container Runtime。开发者将容器镜像作为函数的交付物，通过HTTP协议和函数计算系统交互。使用Custom Container Runtime简化了以下场景： • 低成本迁移，无需修改代码或是重新编译二进制、共享对象（*.so），保持开发和线上环境一致。 • 解压前镜像大小最大支持1 GB，避免代码和依赖分离，简化分发和部署。 • 容器镜像天然的分层缓存，提高代码上传和拉取效率。 • 标准可复现的第三方库引用、分享、构建、代码上传、存储和版本管理，丰富的开源生态CI/CD交付体验。	2020-09-20	• Custom Container简介 • 事件请求处理程序（Event Handler） • HTTP请求处理程序（HTTP Handler） • 创建Custom Container函数
链路追踪	函数计算的链路追踪功能可以串联整个调用链，包含以下核心功能： • 自动记录函数计算内部关键步骤耗时时间。 • 串联上游服务：如果请求Header中带有链路上下文信息，则函数计算会根据上下文创建子链路。 • 串联下游服务：函数计算会将链路上下文传入到函数Context中，帮助您追踪函数内部调用链路。 • 查看应用拓扑。 • 查看错误接口执行情况，定位错误原因。	2020-09-18	• 链路追踪简介 • 配置链路追踪

2020年08月

功能名称	功能描述	发布时间	相关文档
性能实例	函数资源支持更大规格的实例（即性能实例），降低诸多限制给您带来的影响，承诺更明确的资源大小，适配更多的使用场景。	2020-08-31	• 实例类型及使用模式 • 设置实例类型 • 管理函数 • 使用控制台创建函数

2019年10月

功能名称	功能描述	发布时间	相关文档
自定义运行时	函数计算最新支持自定义运行时，完全由您定义函数的运行环境。您既可以基于自定义运行环境定制个性化语言的执行环境，又可以将现有的Web应用不做任何改造，一键迁移至函数计算，轻松构建弹性高可用的Serverless Web应用。	2019-10-01	Custom Runtime用户手册

功能名称	功能描述	发布时间	相关文档
预付费（包年包月）计量模式	函数计算新增预付费（包年包月）计量模式，单价相较于后付费降低70%。负载中稳定部分使用预付费，单价更低；负载中弹性部分使用后付费，资源利用率更高。结合多层次，多维度的资源使用指标，您可以分门别类的追踪应用的资源使用情况，轻松预估资源用量，大幅降低上云开销。	2019-10-01	函数计算成本优化最佳实践

2019年09月

功能名称	功能描述	发布时间	相关文档
预留实例	预留实例功能将函数实例的分配和释放完全交由您管理。常驻的执行环境彻底消除实例冷启动带来的延时。您可以同时使用预留和按量实例。业界创新的不同类型实例间毫秒级别的伸缩能力，提供较高性能的同时保持很高的资源利用率，性能和成本可以兼得。	2019-09-12	• 弹性管理（含预留模式） • 函数计算冷启动优化最佳实践 • 同步调用

2019年07月

功能名称	功能描述	发布时间	相关文档
CDN回源至函数计算	函数计算联合CDN发布CDN回源至函数计算的功能。您在函数计算搭建的应用可一键部署到CDN，对Serverless云原生应用进行加速。此功能的发布完善了基于函数计算搭建弹性高可用云原生Serverless Web应用场景，简化您的操作，提升您的使用体验。	2019-07-07	CDN加速函数计算源站

2019年05月

功能名称	功能描述	发布时间	相关文档
Java HTTP触发器功能	函数计算重磅推出Java HTTP触发器功能，基于Java HTTP触发器可以完成以下操作： • 可以将传统的Java Web应用无缝迁移至函数计算平台，例如Spring、SpringBoot、Struts2等。 • 可以基于Java Runtime快速搭建Cloud Native Web应用。 Java HTTP触发器功能的发布为Java编程爱好者搭建高可用高弹性Web应用提供极大便利。	2019-05-21	代码开发概述

资源编排工具Funcraft发布	• 添加了用户体验改进计划。
	• 优化了*template.yml*语法校验时的错误提示。
	• 添加了当**local invoke**不提供**invokeName**参数时，默认调用第一个函数。
	• 支持自定义PYTHONUSERBASE环境变量。
	• 问题修复：
	  ◦ 修复本地调试HTTP触发器时，二进制数据内容缺失的Bug。
	  ◦ 修复自定义域名语法校验没生效的Bug，并兼容了旧语法。
	  ◦ 配置了Service Role，子用户依旧需要AliyunRAMFullAccess权限的Bug。
		2019-05-21	无

2019年04月

功能名称	功能描述	发布时间	相关文档
支持C#开发环境	函数计算正式支持C#开发环境，并提供同步、异步调用接口，您编写C#代码可以实现各种自定义功能。例如结合HTTP触发器可以快速开发Restful Web API和云原生Web应用；结合定时触发器可以实现自动化运维；结合丰富的事件源服务可以赋予事件源计算的能力。C#开发环境的支持丰富了在函数计算上可运行的编程语言类型，为广大开发者带来了便利。	2019-04-11	代码开发概述

2018年12月

功能名称	功能描述	发布时间	相关文档
表格存储触发器	函数计算推出表格存储触发器，通过表格存储触发器，当您的表格变更时会实时触发函数计算对数据进行自定义处理。使用表格存储触发器，您可以轻松实现对数据的计算、清洗等操作，有效减少配置步骤，大大提升您的使用体验。	2018-12-29	触发器简介
版本管理	函数计算重磅推出版本管理功能，实现生产环境和开发环境的代码隔离。函数计算提供服务级别的版本控制，支持您为服务发布一个或多个版本。同时可以通过别名指向不同的版本，实现新功能的灰度发布。版本管理功能帮助您分离生产环境和开发环境，有效降低您的运维成本，是函数计算的一次重要升级。	2018-12-29	管理版本

2018年11月

功能名称	功能描述	发布时间	相关文档
MNS主题触发器	函数计算推出MNS主题触发器，您可以通过函数对发布在主题上的消息进行自定义处理，使用MNS触发器后，不需要再去轮询MNS内的消息情况，只需要简单的配置，就可以对消息进行丰富的处理，减少操作步骤，大大提升您的使用体验。	2018-11-30	触发器简介
本地调试工具 `fun local`	函数计算推出本地调试工具Funcraft，Funcraft可以将函数算中的函数在本地完全模拟运行，并提供单步调试的功能，旨在弥补函数计算相对于传统应用开发、调试体验上的短板，并为您提供一种排查线上问题的新途径。`fun local` 在本地开发、本地调试上添加了大量的新特性，例如： • 支持本地运行函数。 • 支持本地单步调试函数。 • 支持本地事件触发函数。 • 单步调试时支持展示IDE配置等。 方便您在本地调试，是函数计算工具链上的重要进展。	2018-11-30	无

2018年09月

功能名称	功能描述	发布时间	相关文档

功能名称	功能描述	发布时间	相关文档
函数计算支持自定义域名	函数计算推出自定义域名功能，配合HTTP触发器使用，为您搭建Serverless Web应用提供完美解决方案。搭建Web应用后端是函数计算主要的应用场景，推出HTTP触发器功能后，由于无法绑定自定义域名，您就很难将Web应用迁移到函数计算上来。函数计算支持自定义域名后，仅需两步操作即可绑定自定义域名，解决了您的迁移痛点，大大提升了您搭建Serverless Web应用的使用体验。	2018-09-20	配置自定义域名
云端集成开发环境WebIDE	函数计算WebIDE是一种云端的集成开发环境，您可以在浏览器里编写、运行和调试函数计算代码。它包括代码编辑器、文件树、运行调试器和命令终端，并且内置了fcli、Funcraft等开发工具。您编辑完函数以后借助这些命令工具，将函数发布到函数计算平台执行。从而您能在云端完成开发、调试、部署、运行整个闭环，同时也解决了本地环境差异和配置繁琐的问题。	2018-09-20	功能概览

2018年08月

功能名称	功能描述	发布时间	相关文档
NAS存储挂接	函数计算可以挂接NAS存储，解决您在代码执行空间的受限问题，多个函数可以共用一个NAS空间，实现文件共享。	2018-08-31	配置NAS文件系统
支持PHP运行时	函数计算支持PHP开发环境，允许您使用PHP语言编写函数，并提供两种函数接口： • 普通函数接口 • 针对HTTP触发器请求的函数接口 PHP开发者在函数计算平台上可以充分发挥PHP的能力，进行数据、日志分析，执行任务脚本、搭建Web网站开发等操作。在编写函数的体验上，充分吸取其他语言开发环境的反馈，使得PHP开发简单易上手，在第三方扩展和第三方包的使用上也给予开发者较大的自由空间。	2018-08-31	代码开发概述

2018年07月

功能名称	功能描述	发布时间	相关文档

Funcraft 2.1.0版本发布	• 为API Gateway添加了OpenId的支持，并支持Description描述。
	• 优化日志服务创建的体验
	  ◦ 因为创建过程为异步，并承诺60秒可用，因此加入了重试机制，提高使用体验。
	  ◦ 创建Logstore时，如果没有默认索引，会自动为您创建与控制台默认配置相同的索引。
	• 优化代码包的压缩，主要三个措施：
	  ◦ 提高压缩水平。
	  ◦ 忽略*.git*、*.svn*等版本控制的目录。
	  ◦ 忽略*.env*文件，避免上传隐私信息。
	• 提供了全平台的Funcraft的可执行文件。
		2018-07-10	无

功能名称	功能描述	发布时间	相关文档
控制台代码上传加速	原来在函数计算控制台只能上传最大5 MB的代码包，通过优化控制台前端上传逻辑，现在可以支持上传最大50 MB的代码包，同时还优化了上传速度，让您在函数计算控制台开发应用更加便捷。	2018-07-06	无

2018年06月

功能名称	功能描述	发布时间	相关文档
无服务器应用模型（Serverless Application Model）和开发工具	Funcraft是一个用于支持Serverless应用部署的工具，能帮助您便捷地管理函数计算、API网关、日志服务等资源。其通过一个资源配置文件（*template.yml*），协助您进行开发、构建、部署操作。本次发布新增如下功能： • 兼容ROS语法的Serverless Application Model，定义了规格说明书，并针对规则说明对配置文件进行语法校验。 • 新增支持VPC属性、环境变量属性、日志服务属性、角色属性和Policies属性。 • 当使用默认角色时，会为您自动生成日志、触发器等服务所需权限，并可以通过Policies属性进一步自定义。 • 新增支持日志触发器和HTTP触发器。 • 新增支持创建表格存储的表，解决了之前OTS Instance只能控制台创建的问题。 • 将敏感信息从描述文件中分离，新增 `fun config` 配置向导，生成配置文件兼容fcli。支持ENV和环境变量定义AccessKey ID、AccessKey Secret和Endpoint。 • 支持更多的API网关参数配置。 • 解决了 `fun package` 打包.unix文件权限丢失的问题。 • CodeUri支持FILE、DIR、ZIP和OSS-Bucket等形式。 • 添加了Node版本检查。	2018-06-29	功能概览
Go SDK发布	发布Golang SDK，让Golang 开发者能调用函数计算服务API管理或调用函数。	2018-06-29	无
访问用户VPC内的云资源	专有网络VPC是您基于阿里云创建的自定义私有网络。函数计算通过挂接用户弹性网卡（ENI）的方式，能安全、合规的访问用户VPC中的RDS，ECS等云资源。这样函数能够和您已有的云服务交互，大大拓展无服务器应用的场景。	2018-06-29	配置网络
CDN触发器	• CDN在预热（CachedObjectsPushed）和刷新（CachedObjectsRefreshed）用户数据后，通过触发器通知用户函数。避免您不断轮询列表查询最新状态，可以及时得知资源预热刷新的状态并进行下一步处理。 • 当在CDN上发现违禁内容（CachedObjectsBlocked）时，通过触发器通知用户函数直接去源站删除资源。您不需要等待CDN团队响应，可以及时去源站删除资源。 • 日志文件生成后（LogFileCreated），通过触发器通知用户函数处理日志。您不需要长时间等待日志，可以及时转存或处理日志。 • 当某加速域名被停用（CdnDomainStopped）或者被启用（CdnDomainStarted），通过触发器通知用户函数及时做出相应处理。	2018-06-28	触发器简介

2018年05月

功能名称	功能描述	发布时间	相关文档
HTTP触发器Python版本	函数计算推出HTTP触发器Python版本，方便您通过HTTP请求来调用函数，适合互联网行业、AI行业、IoT行业的开发者调用函数执行。	2018-05-21	触发器简介

2018年04月

功能名称	功能描述	发布时间	相关文档
HTTP触发器	函数计算推出HTTP触发器，方便Web使用者通过HTTP请求即可触发函数的执行，方便开发者进行请求调试。	2018-04-30	HTTP触发器概述
环境变量	您可以设置函数里的环境变量，有利于敏感信息的存储和读取，方便在函数中引用环境变量信息。	2018-04-28	环境变量

2018年03月

功能名称	功能描述	发布时间	相关文档
澳大利亚悉尼海外节点	函数计算正式推出澳大利亚悉尼海外节点，函数计算团队会持续推出更多海外节点，为您的业务提供更多地域选择。	2018-03-27	无
时间触发器功能	函数计算重磅推出时间触发器，方便您通过时间来定制业务的执行顺序，大大拓展您的业务范围。	2018-03-16	触发器简介
大内存3 GB的功能	函数计算推出大内存3 GB的功能，从底层支持您使用更大的执行内存，这样的一个功能推出，极大方便对内存要求高的业务选择，您再也不用担心内存不够用的问题。	2018-03-16	什么是函数计算

2018年02月

功能名称	功能描述	发布时间	相关文档
多款OSS实用模板	函数计算推出多款OSS实用模板，可以实现以下功能： • 计算OSS文件的MD 5值 • 备份OSS增量数据 • 针对文件打包并下载等功能模板 大大降低了开发者的使用难度。	2018-02-03	应用场景

2018年01月

功能名称	功能描述	发布时间	相关文档
Node.js 8语言编程	函数计算推出最新Node.js 8语言编程环境，目前已经推出了Node.js 4.4，Node.js 6.0，持续改善开发者体验，Node.js 8版本提供一系列新功能和性能改进。	2018-01-17	代码开发概述

2017年12月

功能名称	功能描述	发布时间	相关文档

功能名称	功能描述	发布时间	相关文档
与IoT产品的对接	函数计算打通和IoT产品对接，通过IoT产品控制台可以配置函数计算作为后端处理，可以解决物联网，智能家居存在的问题： • 数据不能即时处理 • 较重的技术架构 • 无法实时弹性扩容 • 高额成本支出 IoT和函数计算打通，函数计算作为后端处理，可以构建无服务器架构的应用场景，以及高弹性、低成本的场景。	2017-12-23	通信方式概述
全新版本控制台	函数计算正式推出全新版本控制台，新版控制台提供计量数据和各种分类导航，整体界面紧凑简洁，编辑页面使用抽屉推出方式，减少大量界面交互跳转。让您在一个页面里操作多个功能。新推出的控制台的特性如下： • 显示业务调用量 • 显示最新的触发器 • 显示最新模板详情 • 显示产品更新动态 • 抽屉式的页面 • 内容更丰富紧凑	2017-12-22	无
OSS控制台增加函数计算处理事件入口	OSS控制台新增函数计算处理事件入口，真正双向联通存储和计算，通过函数计算可以管理某个函数下的触发器事件，通过OSS的某Bucket可以管理该Bucket下的定义的事件，让两个产品真正融为一体，大大提升您的使用体验。	2017-12-15	部署程序包
华南1区开通	函数计算开通华南1地域，增加深圳节点。	2017-12-11	无

2017年11月

功能名称	功能描述	发布时间	相关文档
Python 3.0编译环境	函数计算正式推出Python 3.0编译环境，提升开发者体验。	2017-11-16	代码开发概述

2017年10月

功能名称	功能描述	发布时间	相关文档
日志服务触发器	函数计算推出了日志服务触发器，新增以下功能： • 可以无缝对接日志服务。 • 可以作为日志服务的后端处理。 • 可以对日志进行清洗，压缩，提取等。 • 可以在日志服务来设置定时处理事件。	2017-10-10	触发器简介

2017年09月

功能名称	功能描述	发布时间	相关文档
集成表格存储	函数计算打通了和表格存储访问，可以通过表格存储的表格增删改查等事件来触发函数计算的执行，真正完成智能数据的处理。	2017-09-30	使用函数计算

功能名称	功能描述	发布时间	相关文档
阿里云函数计算商业化	函数计算是一个无服务器的全托管的运行环境，您只需编写代码并设置运行的条件，就可以弹性、安全地运行。函数计算会自行维护服务器资源，网络资源，以及消息分发和负载均衡等功能。其特点如下： • 能大规模并行执行代码。 • 无需配置服务器、安装软件、部署容器。 • 容量自动配置、自动扩展机制，松散耦合。 • 事件触发运行代码方式。 • 按运行计费，不执行0费用。	2017-09-27	无
华东1区开通	阿里云函数计算开通华东1地域。	2017-09-25	无
4大实用模板	函数计算强力推出4大实用模板，例如打标签、爬虫算法、API网关对接返还串等多个模板，降低您的使用门槛，协助开发者更便捷的实现业务逻辑。	2017-09-19	无

2017年07月

功能名称	功能描述	发布时间	相关文档
3套可定制化OSS模板代码	函数计算推出可定制化OSS模板代码示例，方便开发者通过示例快速上手，降低学习成本。	2017-07-29	无
Python 2.7的编译	函数计算正式推出Python 2.7编译环境，提升开发者体验。	2017-07-04	代码开发概述

2017年06月

功能名称	功能描述	发布时间	相关文档
接入API网关	如果您使用API网关，您就可以直接使用函数计算作为后端服务。	2017-06-05	无

2017年05月

功能名称	功能描述	发布时间	相关文档
函数服务增加获取当前版本代码API	新增GetFunctionCode API，方便您进行错误分析。	2017-05-26	无
新版SDK	方便您在Java程序Node.js中调用函数服务。	2017-05-09	无

2017年04月

功能名称	功能描述	发布时间	相关文档
函数服务	事件驱动的Serverless服务，毫秒级弹性伸缩。	2017-04-27	无

2.客户案例

2.1. 世纪联华

本文介绍如何通过函数计算、API网关、日志服务及表格存储云服务，再配合Serverless架构的快速扩容、弹性高可用的特性，轻松解决世纪联华流量突增及服务器需求过大的问题。

客户介绍

世纪联华是一个超过170余家，具有全国网点布局，在快速消费品连锁零售行业具备较高地位的新零售连锁超市。

世纪联华经过多年的发展，业态包括大卖场、综合超市和标准超市，公司定位为超市和精品百货，拥有“世纪联华”、“联华”等超市品牌。

客户痛点

- 用户较集中：世纪联华粉丝较多，每月推出的优惠消息会有大量会员同时登录抢券，突发点太集中，导致服务无法正常运行。
- 服务器需求过大：超市会员管理业务密集，尽管已经为支撑定时线上抢购促销活动堆积了大量服务器，但在业务高峰期还是会对服务器造成很大冲击。
- 系统过载：双十一这样的大型促销活动，可能会遇到繁琐复杂的服务器管理工作、用户浮动导致的系统过载等情况，世纪联华需要找到一种弹性稳定的系统架构来平滑支撑。

解决方案



Serverless架构具有快速扩容、弹性高可用的特性，可以轻松应对突发流量。 使用阿里云表格存储代替传统的关系型数据库，通过函数计算读写表格服务中的数据，快速返回处理结果给前端用户。

与传统方案临时增设服务器相比较，通过函数计算、表格存储及API网关云服务极大地简化了运维。

- 函数计算：具有弹性扩容的优势，根据请求量动态分配执行环境，部署上线简单。
- 表格存储：具有访问更快，吞吐量更强，不需要额外增加机器的优势。
- API网关：可以控制访问量，导出API文档，非常方便。

使用效果

- 增加收入：联华超市在阿里云的技术护航下，在双十一大型促销活动中，三天大促交易额创5.5亿新高。
- 减少工作量：阿里云的函数计算减少了技术人员的大量工作。 函数计算是事件驱动的全托管计算服务。使用者无需管理服务器等基础设施，只需要编写代码上传，系统不仅将计算资源准备好，还提供日志查询、性能监控、报警等功能。 所有的数据和业务上云，减轻的不止是使用者的心理压力，还有工作量。如果没有函数计算，联华超市在双十一大型促销活动中，如果只靠扩展机器来支撑大体量的流量和业务，将无法保障活动正常进行。阿里云解决了扩容问题，极大地提高了数据储备能力。
- 减弱对技术人员的依赖：阿里云提供前沿技术，商超的开发人员无需自己钻研算法，会使用阿里云的相关工具即可，降低了研发投入和成本。

2.2. 捷旅假期

本文以捷旅假期为例，介绍如何通过函数计算、

消息队列RocketMQ版
及对象存储等云服务，处理瞬时并发消息等问题。

客户介绍

深圳市捷旅国际旅行社有限公司（简称“深捷旅”），成立18年来专注于酒店行业的客房分销业务以及酒店预订信息化水平建设，通过“捷旅假期”品牌打造了业内领先的酒店B2B分销平台。深捷旅一直秉承 “便捷、实惠、高效”的服务理念，围绕打造“中国最好的B2B旅游生态圈”目标前进。目前，全球合作的旅游分销渠道超过1万家，拥有全球超过60万家星级酒店资源，覆盖了全球300多个城市。

深捷旅与业内旅游头部OTA企业的多年深耕合作，为同程旅游、去哪儿、途牛旅游网、 Expedia、Ctrip、Agoda、驴妈妈旅游网、马蜂窝旅游、国旅运通、大唐商旅等企业提供完美的酒店及票务服务。

客户痛点

捷旅需要对接的酒店超过60万家，每天新增的信息超过千万条、信息并发脉冲性强且时效周期短，捷旅平台处理瞬时并发消息压力大。因此捷旅平台期望具备以下优势，缓解并发处理消息的压力。

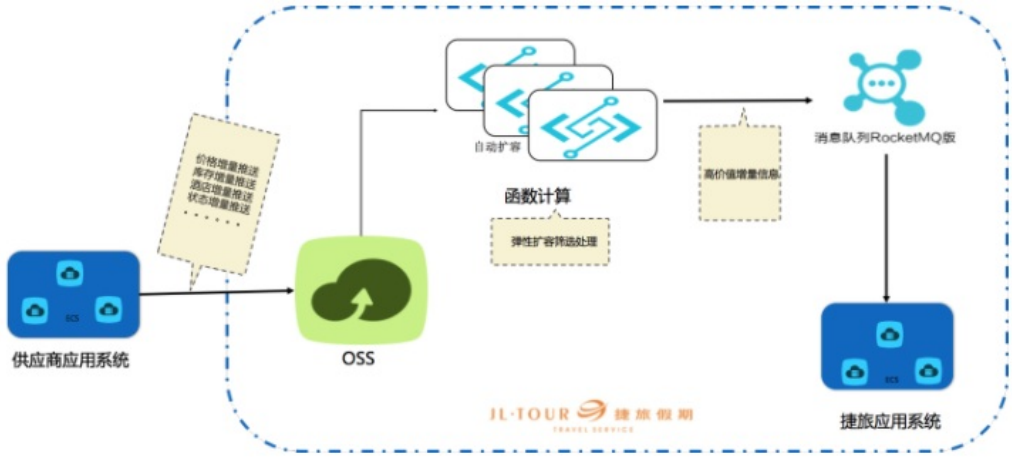
- 并发处理能力：同时处理超过10万条的峰值消息。
- 弹缩处理能力：根据消息量的多少进行自动化毫秒级的扩缩容操作，使用成本按所需的实际资源付费。
- 支持多种触发方式：支持对象存储服务、消息存储等触发。
- 支持多种编辑语言：支持Python、Go及Java等语言。
- 运维监控能力：支持快速部署更新及资源使用的实时监控，日志分析集成与报警能力。

解决方案

通过函数计算、

消息队列RocketMQ版
及对象存储等云服务解决了捷旅平台的痛点：

- 函数计算支持监听多种数据源：
  - 监控处理业务量的变化，快速进行自适应的扩缩容操作。
  - 监控毫秒级的扩容，获得线性增长的业务处理能力。
- 函数计算支持多种编程语言，无需用户改变编程习惯。
- 函数计算支持便捷的部署、资源使用的实时监控、日志分析集成与报警能力。



使用效果

- 业务稳健：函数计算可根据业务量进行扩缩容，在函数计算处理的帮助下，解决了之前按峰值配置资源方式的业务稳定性问题。
- 运维简单：运维发布有多种工具辅助进行。使用函数计算后，解决了资源扩缩容管理问题，减少了深捷旅开发运维人员的人力投入。
- 降低成本：利用函数计算的按需付费特性，在合适的规格运行下，使得整个资源得到了高效利用，不为闲置资源付费，降低了深捷旅的总体使用成本。

2.3. 石墨文档

本文以石墨文档为例，介绍如何借助函数计算毫秒级别的资源伸缩能力，解决计算资源扩容问题，降低服务器成本，提高工作效率。

客户介绍

石墨文档是中国第一款支持云端协作的企业办公服务软件，2014年成立至今，石墨文档已累计为超过20万家企业提供了产品和服务，其功能类比于Google docs、Quip等软件，可以实现多成员多终端的在线实时协作，同步响应速度达到毫秒级，是团队协作的最佳选择。 石墨文档还是一款具有中国式美感的科技产品，2015年进入极客公园最佳互联网创新产品50强。

客户痛点

为了支持实时协作编写，技术团队对Operational Transformation算法进行深入地探索和修改，将维度从一维拓展到二维，解决了一段文字被多个客户端同时修改所引起的冲突合并问题。但石墨文档仍面临以下挑战：

- 多用户的实时修改对服务器的要求比较高。
- 用户敲击键盘输入一个文字只需要几毫秒，当多用户在石墨文档上同时编写文档时，很容易出现在短时间段内数据分布不均的情况。

石墨文档需要一套弹性高可用的服务来实时处理文档编辑冲突，该服务需满足以下需求：

- 低成本。
- 负载突增时平滑处理峰值负载，满足文档在毫秒级别的同步响应。

解决方案

函数计算是阿里云提供的Serverless计算平台，可以根据请求量动态分配执行环境，毫秒级调度计算资源，在负载高时保持稳定的延时；在负载低时有较高的资源利用率，且只对代码运行时使用的计算资源付费。石墨文档通过函数计算借助公有云Serverless架构搭建文档实时编辑服务，将文档实时协作的逻辑实现为函数，由函数计算的智能调度系统自动分配执行环境，处理多用户同时编写文档带来的峰值负载。函数计算的动态扩缩容能力保障应用的可靠运行。

使用效果

- 降低成本：石墨文档借助函数计算毫秒级别的资源伸缩能力，解决了早晚高峰负载突增的计算资源扩容问题，相较于自建机房维护服务器，提高了资源利用率，减少了闲置资源的浪费，降低了服务器成本。
- 提高效率：石墨文档借助函数计算，开发人员只需集中精力与产品团队合作，无需考虑CPU密集型计算的负载均衡问题，不断提升业务价值，加快项目的迭代与上线的步伐，极大地提高了开发效率和进程稳定性。

2.4. 新浪微博

本文以新浪微博为例，介绍如何使用函数计算的弹性扩容、事件触发及按量付费的功能，解决因业务高峰带来的流量激增等问题。

客户介绍

新浪微博（Sina Weibo）是基于用户关系的社交媒体平台，用户可以通过电脑、手机等多种移动终端接入，以文字、图片、视频等多媒体形式，实现信息的即时分享、传播互动。每个人都可以创作并发布微博，人人都是世界的主角。新浪微博基于公开平台架构，使用户能够公开实时发表内容，通过裂变式传播，让用户之间进行互动并与世界紧密相连。

客户痛点

自2009年8月上线以来，新浪微博的用户一直处于爆发式增长的状态。截止2019年6月，新浪微博月活跃用户数高达4.86亿，平均日活跃用户数为2.11亿。业务的急速增长对微博的原有技术体系造成了极大的冲击，例如：

- 流量激增：一些业务场景例如元旦、春晚、红包飞给微博带来巨大的流量挑战，这些业务场景的主要特点是：瞬间峰值高、持续时间短。每一次峰值事件的互动时间在3小时左右，而明星事件、红包飞等业务，流量经常会增加至原来几倍的瞬间峰值。
- 流量峰谷差大：社交媒体与人们生活作息时间紧密相关，在奔波忙碌的午后，人们专心工作无暇分享生活，流量请求量平缓；在闲暇安逸的傍晚，人们纷纷掏出手机查看当日的新鲜事，流量请求量激增。在生活的一朝一夕背后，微博的负载有着明显的波峰波谷，且峰值相差5倍以上。

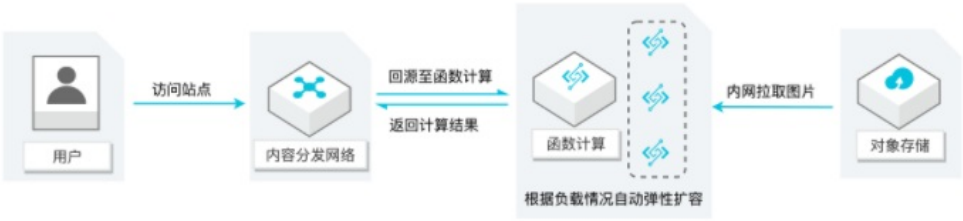
应对流量激增问题，新浪微博开发者的传统应对手段，主要通过以下几种方式：

- 提前申请足够的设备保证冗余。

- 降级非核心及周边的业务。

但以上解决方案需要提前预知相关IT成本，造成业务负载饱和度不一及扩缩容流程繁琐且周期长等问题。如何在用户量飞速增长、热点事件流量激增及请求有明显波峰波谷的情况下，既不影响用户体验，又不增加服务器成本投入是新浪微博开发人员的当务之急。

解决方案



公有云Serverless架构具有应对爆发式峰值流量的优点。函数计算是阿里云提供的Serverless计算平台，函数计算可以根据请求量动态分配执行环境，毫秒级调度计算资源，确保在负载高时保持稳定的延时，在负载低时有较高的资源利用率，且只会对代码运行时使用的计算资源付费。函数计算还可以与对象存储服务无缝集成，可以方便地对存储在对象存储中的图片进行实时处理。

新浪微博使用函数计算的弹性扩容、事件触发、按量付费的特性部署图片处理业务，将用户上传的图片存储到对象存储中，编写函数实现个性化的图片处理。当微博的用户通过客户端获取图片时，请求通过阿里云CDN回源到函数计算，函数从对象存储中下载原图，根据客户端类型实时处理成预期规格的图片，并将结果图片返回。

使用效果

函数计算可以毫秒级伸缩计算资源确保应用在热点事件发生时仍能稳定地运行，使用户体验不受访问次数的影响。

通过函数计算运行图片处理服务，新浪微博实现了持续的成本节省。无需再为平滑处理业务高峰带来的流量激增而提前预留大量闲置机器资源，同时开发人员无需花费精力管理及维护基础设施，只需集中精力与产品团队合作提升业务价值。

随着新浪微博的活跃用户数不断增加，新浪微博业务规模不断扩张，函数计算可以自动弹性地分配更多执行环境以支撑微博业务的持续发展。

2.5. 闲鱼

本文介绍如何通过函数计算，解决闲鱼研发耦合严重和运维复杂等问题。

客户介绍

闲鱼是依托阿里电商体系的前台型业务，有非常独特的业务特点和用户诉求，在底层依托阿里系统的同时，在表现层和业务层需要探索适合闲鱼的、并且更加快速灵活的研发体系。

从2018年开始，闲鱼架构负责人带领闲鱼技术团队布局Serverless。颠覆性创新往往是从边缘性的地方出现，而向云原生化及Serverless化升级，对于闲鱼是一条全新的路，走出了这条路，对于很多做线上交易的公司有着巨大的借鉴意义。

客户痛点

- 客户端交互层、服务端业务胶水层、领域层边界划分不清晰，导致很小的业务需求就需要整条链路的同学参与，协同成本高，开发调试周期长。
- 服务端存在巨型应用，研发耦合、发布耦合和运维耦合严重，系统稳定性也受到很大挑战，单个业务问题往往会影响整个应用。
- 运维成本极高。为了保障业务的稳定性和可用性，阿里巴巴对每一个应用上线都有相应的规范和规则。哪怕是一个很小的内部应用，一天可能只有一两个访问量，上线也需要遵守既有的规范，这势必会消耗一些固定资源。单个应用消耗的资源可能很有限，但所有应用消耗的资源累积起来也是一个不小的数字。而对于巨型应用，由于影响面巨大，发布时要有更加严格的流程和步骤，一次发布至少要耗时6小时，运维成本极高。

解决方案

Serverless技术使云端一体化研发成为可能，很多小业务需求的协同成本可以大大降低。另一方面，Serverless使业务胶水层的巨型应用，有了比微服务更加合理的拆分方式。

云原生及Serverless这些新技术的出现，可以使应用运维能力下沉，打破传统巨型应用的成本、稳定、质量相互制约的瓶颈。闲鱼在落地新技术的过程中，先围绕Flutter重点攻坚了Flutter混合工程体系、高性能组件库。然后围绕Serverless重点攻坚云端一体化研发体系、服务端业务组装层架构体系。

闲鱼客户端基于Flutter进行架构演进与创新，通过Flutter统一Android和iOS双端提升研发效能之后，希望通过Flutter加Serverless解决各角色间存在的大量的协同问题，从而解决整体研发效率低，移动端离业务越来越远，服务端没有时间做底层领域沉淀的问题。通过Serverless的引入，可以看到闲鱼整体研发效率得到明显的提升。

使用效果

- 云端编程模型一体化框架Nexus API。
  这个框架的目标是使Flutter、FaaS的编程模型统一，打通UI、交互、数据和逻辑。闲鱼架构负责人提到，一开始说要做Flutter+FaaS一体化的时候，他们对一体化这三个字的认知相对比较模糊，只是知道端侧的同学可以用Dart这门语言来写FaaS函数，这其实还停留在语言上的一体化。对于FaaS所能做的事，也仅仅停留在前端实施已久的BFF层面。
- CLI开发工具标准化。
  云端一体化开发时，通过CLI命令行工具屏蔽FaaS开发的一些细节，使客户端开发FaaS时的开发体验标准化，符合客户端同学的本地开发习惯。
- 基础服务BaaS化。
  过去两年，闲鱼在逐渐简化基础服务能力，如对象存储、消息、搜索。同时，建设业务领域层服务的元数据中心，这些简化的基础服务能力，再加上已有的业务领域层服务，使客户端同学可以快速组装业务。
- 云端工程一体化。
  闲鱼在成功引入Flutter后，在端侧形成了以Flutter为主、H5为辅的跨端研发体系，使传统的Android和iOS的两端研发，合并成一端。在端上的生产力得到释放时，闲鱼发现端的同学有机会向下层走一点，使服务端面向简单的数据组装逻辑，由端的同学一人闭环完成，这套模式尤其适用于一些小业务的需求。
- 传统巨型应用的Serverless化改造。
  Serverless与业务胶水层的特点很匹配，非常适用于解决胶水层的传统巨型应用的拆分，这也是闲鱼正在攻坚的下一个难题。

从最开始不被外界看好，甚至被调侃为“咸鱼”，到如今实现了千万DAU，盘活了一个万亿级市场，闲鱼的出现，无论是对前端的电商生态，还是用户在互联网上的生活形式，都产生了重要的影响。

为了支撑起闲鱼万亿的交易规模，技术团队正在紧锣密鼓地进行传统巨型应用的Serverless化改造。

2.6. 语雀

本文介绍如何通过函数计算，解决语雀CPU密集场景下，进程被阻塞等问题。

客户介绍

语雀是一个专业的云端知识库，用于团队的文档协作。现在已经是阿里巴巴员工进行文档编写和知识沉淀的标配，并于2018年开始对外提供服务。

客户痛点

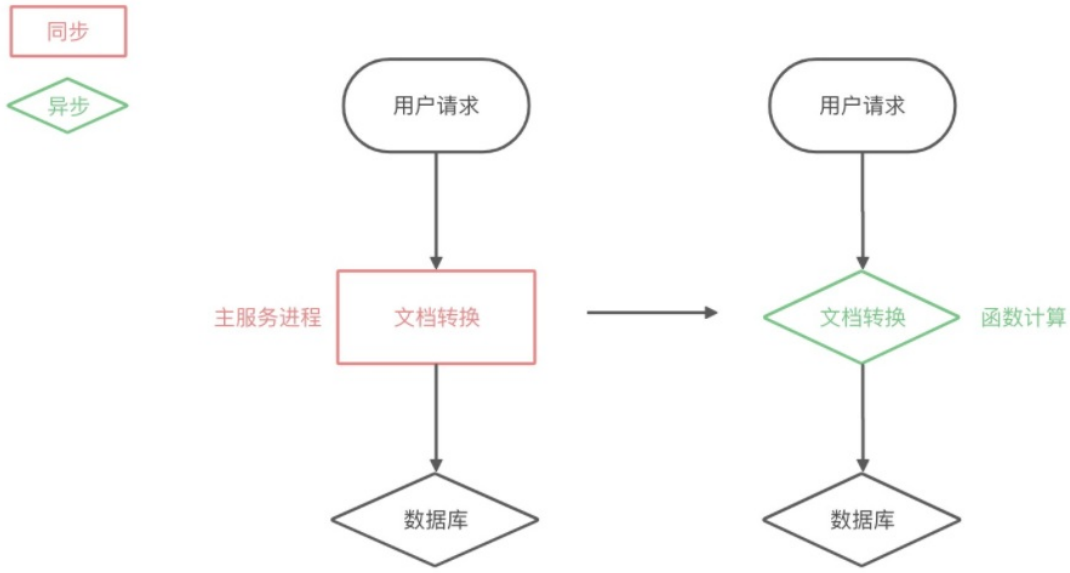
语雀是一个复杂的Web应用，也是一个典型的数据密集型应用（Data-Intensive Application），背后依赖了大量的数据库等云服务。语雀服务端是Node.js技术栈。Node具有单线程（single-threaded）、非阻塞（non-blocking）、异步（asynchronously programming）等特性，这些特性一方面非常适合于构建可扩展的网络应用，用来实现Web服务这类I/O密集型的应用。但是，Node对CPU密集型的场景不够友好，一旦有任何阻塞进程的方法被执行，整个进程就被阻塞。

像语雀这样用Node实现整个服务端逻辑的应用，很难保证不会出现一些场景可能会消耗大量CPU甚至是死循环阻塞进程的，以Markdown转换举例，由于用户的输入无法穷举，总有各种可能让转换代码进入到一个低效甚至是死循环的场景之中。在Node刚出世的年代，很难给这些问题找到完美的解决办法。即便是Java等基于线程并发模型的语言，也无法完美解决这种场景。毕竟CPU对于Web应用来说是非常重要的资源。而随着基础设置越来越完善，当函数计算出现时，Node最大的短板看起来有了一个比较完美的解决方案。

解决方案

把函数计算引入之后，语雀将那些CPU密集型、存在不稳定因素的操作统统放到函数计算服务中去执行，而语雀的主服务再次回归到了I/O密集型应用模型，又可以愉快的享受Node给语雀带来的高效研发福利。

以语雀中遇到的一个实际场景来举例，您上传了一些HTML或者Markdown格式的文档内容，语雀需要将其转换成为语雀自己的文档格式。在绝大部分情况下，解析您输入的内容都很快，然而依然存在某些无法预料到的场景会触发解析器的故障而导致死循环的出现。甚至语雀不太敢升级Markdown解析库和相关插件以免引入更多的问题。但是随着函数计算的引入，语雀将这个消耗CPU的转换逻辑部署到函数计算上，语雀的主服务稳定性不会再被影响。



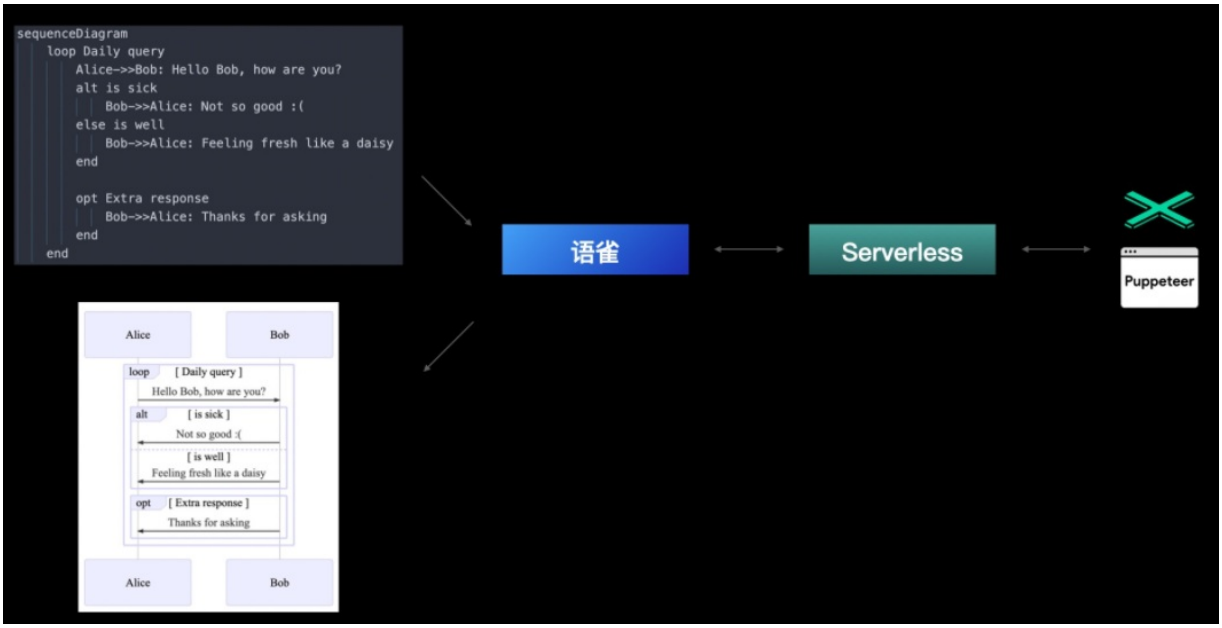
语雀支持使用各种代码形式来绘图，包括Plantuml、公式、Mermaid，还有一些将文档导出成PDF、图片等功能。这些场景有两个特点：

- 依赖于一些复杂的应用软件，例如Puppeteer、Graphviz等。
- 可能需要执行您输入的内容。

支持这类场景看似简单，通过process.exec子进程调用一下即可。但是当语雀想把它做成一个稳定的对外服务时。这些复杂的应用软件可能从设计上并没有考虑要长期运行，长期运行时的内存占用、稳定性可能会有一些问题；同时这些应用在被大并发调用时，对CPU的压力非常大。再加上有些场景需要运行用户输入的代码，攻击者通过构建恶意输入，可以在服务器上运行攻击代码，非常危险。

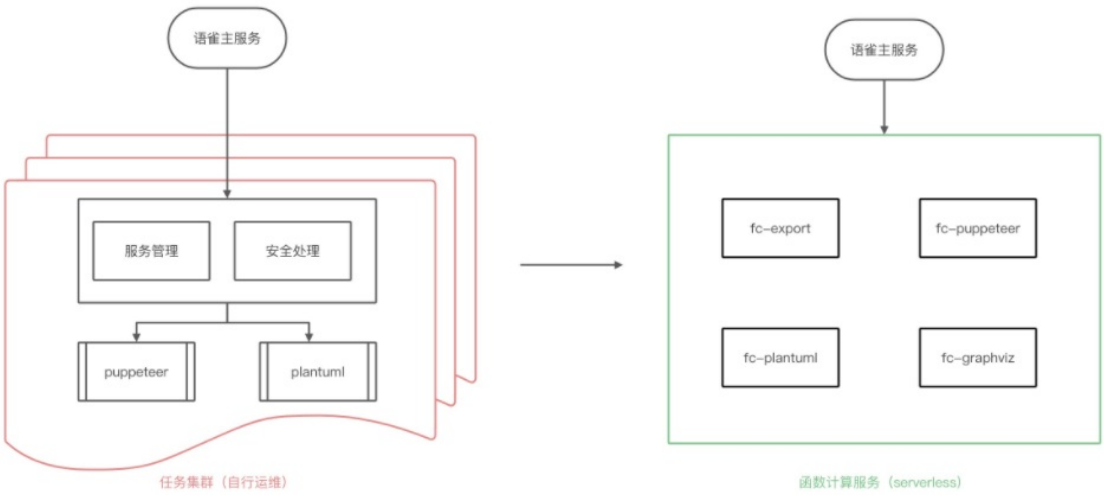
在没有引入函数计算之前，语雀为了支持这些功能，尽管单独分配了一个任务集群，在上面运行这些三方服务，接受主服务的请求来避免影响主服务的稳定性。但是为了解决上面提到的一系列问题还需要付出很大的成本：

- 需要维持一个不小的任务集群，尽管可能大部分时间都用不上那么多资源。
- 需要定时对三方应用软件进行重启，避免长时间运行带来的内存泄露，即便如此有些特殊请求也会造成第三方软件的不稳定。
- 对您的输入进行检测和过滤，防止黑客恶意攻击，而黑客的攻击代码很难完全防住，安全风险依旧很大。



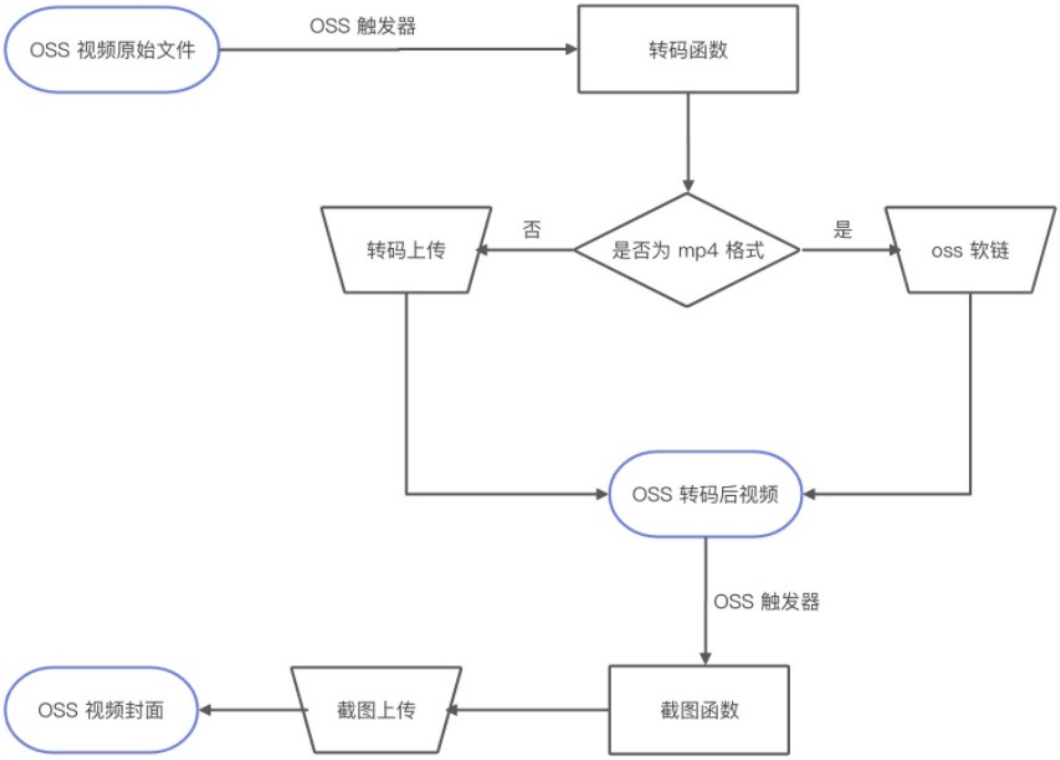
最后语雀将所有的第三方服务都分别部署在函数计算中，将这个任务集群上的功能都拆分成了一系列的函数部署到函数计算。通过函数计算的特点一下解决了上面的所有问题：

- 函数计算的计费模式是按照代码实际运行的CPU时间计费，不需要长期维护一个任务集群了。
- 函数计算上的函数运行时尽管会有一些常驻函数的优化，但是基本不用考虑长期运行带来的一系列问题，且每次调用之间都相互独立，不会互相影响。
- 用户的输入代码是运行在一个沙箱容器中，即便不对用户输入做任何过滤，恶意攻击者也拿不到任何敏感信息，同时也无法进入内部网络执行代码，更加安全。



除了上面提到的这些功能之外，语雀最近还使用OSS加函数计算替换了之前使用的阿里云视频点播服务来进行视频和音频的转码。

由于浏览器可以直接支持播放的音视频格式并不多，大量用户上传的视频想要能够直接在语雀上进行播放需要对它们进行转码，业界一般都是通过FFmpeg来对音视频进行转码的。转码服务也是一个典型的CPU密集型场景，如果要自己搭建视频转码集群会面临大量的资源浪费，而使用阿里云视频点播服务，成本也比较高，而且能够控制的东西也不够多。函数计算直接集成了FFmpeg提供音视频处理能力，并集成到应用中心，配合SLS完善了监控和数据分析。语雀将音视频处理从视频点播服务迁移到函数计算之后，通过优化压缩率、减少不必要的转码等优化，将费用降低至之前的20%。



使用效果

从语雀的实践来看，语雀并没有像SFF一样将Web服务迁移到函数计算之上（SFF模式并不是现在的函数计算架构所擅长的），但是函数计算在语雀整体的架构中对稳定性、安全性和成本控制起到了非常重要的作用。总结下来函数计算非常适合下面几种场景：

- 对于时效性要求不算非常高的CPU密集型操作，分担主服务CPU压力。
- 当做沙箱环境执行用户提交的代码。
- 运行不稳定的三方应用软件服务。
- 需要很强动态伸缩能力的服务。

在引入函数计算之后，语雀现阶段的架构变成了以一个Monolith Application为核心，并将一些独立的功能模块根据使用场景和对能力的要求分别拆分成了Microservices和Serverless架构。应用架构与团队成员组成、业务形态息息相关，但是随着各种云服务与基础设施的完善，语雀可以更自如地选择更合适的架构。

由于Serverless的出现，语雀可以将这些存在安全风险的，消耗大量CPU计算的任务都迁移到函数计算上。它运行在沙箱环境中，不用担心用户的恶意代码造成安全风险，同时将这些CPU密集型的任务从主服务中剥离，避免出现并发时阻塞主服务。按需付费的方式也可以大大节约成本，不需要为低频功能场景部署一个常驻服务。

2.7. 清林云

阿里云函数计算承载了清林云的业务API层、常用应用和自定义应用，再配合使用Serverless工作流，使得整个流程Serverless化，轻松应对清林云高峰流量和集中自动化流运行。

公司介绍

清林云是一家专业的互联网业务咨询公司，为客户提供技术、产品等咨询和服务支持，加速企业数字化转型和新业务拓展， 在多个行业拥有成熟的整体解决方案，经历亿级用户考验。

客户痛点

随着RPA和IFTTT等企业自动化市场的发展，某客户决定转型进入自动化流市场，委托清林云建设类似IFTTT的自动化流平台，使其他公司技术人员和非技术人员都可以在Web和小程序界面以拖拽的方式来编辑业务逻辑。企业自定义或购买应用后，可以按照流程对业务进行逻辑编辑，以更灵活的方式来更改业务策略，实现自动化营销、自动化的用户增长，以及企业工作流的自动化等。但构建过程中遇到了以下难题：

- 平台研发成本高：自动化流的应用结构极其复杂，开发难度高，成本大，涉及资源广。
- 稳定性难以保障：某节点的宕机会导致整个服务的中断。
- 问题定位困难：长时间流的执行丢失，若出现内存泄漏等问题，将难以定位，流状态管理复杂。
- 高阶企业级需求难满足：自动化流的分布式问题难以解决，定时等任务还需要引入外部组件，多租户资源隔离难，同时还会遇到执行流冲突等问题。
- 构建成本高：需要维护大量的服务器资源和队列消息。

解决方案

使用阿里云函数计算承载业务API层、常用应用和自定义应用，使用Tablestore作为后端数据库，再加上Serverless工作流，作为所有自动化流的底层实现，分析类业务接入下游的数据库，日志统一接入日志服务，打通了整个链路，使得整个流程Serverless化，轻松应对高峰流量和集中自动化流运行。

使用效果

通过使用Serverless工作流简化了自动化流应用的开发和维护，极大地减少了开发时间和开发成本，并且直接拥有了阿里云Serverless工作流的成熟模式，使得自动化流上线之初，就拥有了完备的逻辑支持。很多自动化流的执行时间比较集中，阿里云的托管服务使得高峰值运行也能平稳流畅，为客户带来了良好的体验。Serverless工作流的应用模型保障了多租户情况下的资源隔离，自动化流的冲突不复存在，安全性和可靠性大大提升。全套Serverless系统使得成本大幅减少，同时开发效率又成倍提升，很好的支持了新业务的发展。

清林云当前已经上线的系统依然需要访问外部服务，后续的系统将继续结合函数计算和Serverless工作流，实现自定义应用部署到函数计算，为自动化流系统带来更多的业务场景。

2.8. 高德

Serverless作为高德出行的一个核心技术场景，平稳扛住了2020年的十一出行节流量高峰期的考验。值得一提的是，由Serverless支撑的业务在流量高峰期的表现十分优秀，每分钟函数调用量接近两百万次。这再次验证了Serverless基础技术的价值，进一步拓展了技术场景。

公司介绍

2020年的十一出行节期间，高德地图创造了记录——截止2020年10月01日13时27分27秒，高德地图当日活跃用户突破1亿，比2019年10月01日提前3时41分达成此记录。

客户需求

自主出行是高德地图的核心业务，涉及到用户出行相关的功能诉求，承载了高德地图App内最大的用户流量。自主出行核心业务中应用Node FaaS的部分场景，包括主图场景页、路线规划页和导航结束页等。

随着功能的进一步拓展，高德地图从导航工具升级为出行服务平台和生活信息服务入口，进一步拓展了出行相关的生活信息服务场景，带给用户更全面的用户体验。例如场景推荐卡片，旨在根据您的出行意图推荐信息，提升您的出行体验。此功能需具备快速迭代，样式调整高灵活性的能力。因此，将卡片样式模版存放于云端，通过服务下发的形式渲染至客户端无疑为最优选择，可以满足业务快速灵活迭代的目的。

解决方案

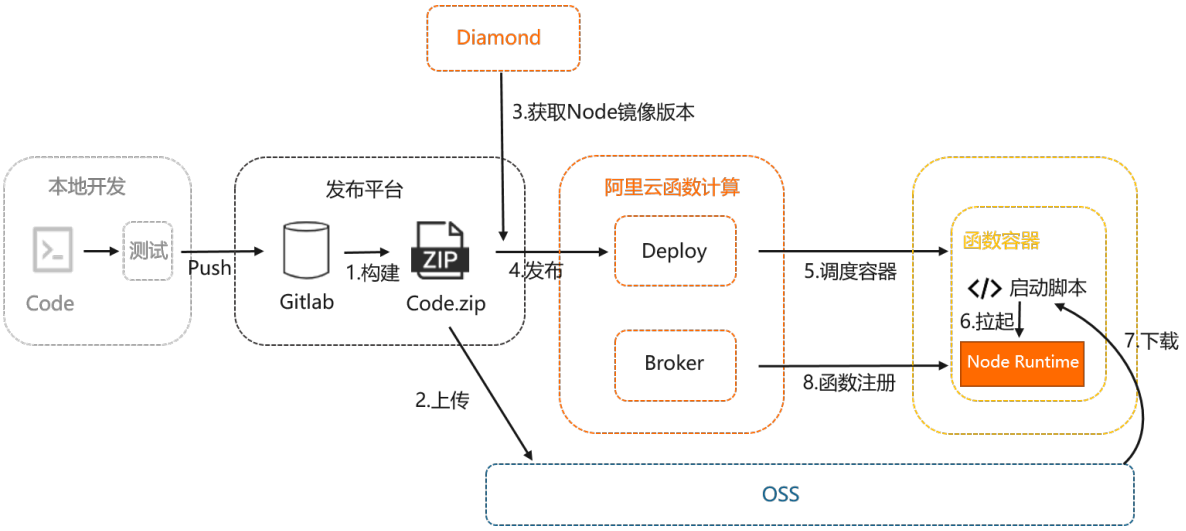
经过方案评估判断，此场景类型属于无状态服务，基于阿里云Serverless成熟的生态，高德最终选择接入Node FaaS（阿里云函数计算）服务能力，出行前端搭建了场景推荐卡片服务。卡片的UI模版获取、数据请求聚合&逻辑处理、拼接生成Schema的能力均在FaaS层得到实现，客户端根据服务下发的Schema直接渲染展示，达到更加轻便灵活的目标。在十一出行节峰值场景中，Serverless整体服务成功率均大于99.99%，总计每分钟触发100W次以上，QPS 2W以上，各场景的服务平均响应时间均在60 ms以下，服务稳定性超出预期。

使用效果

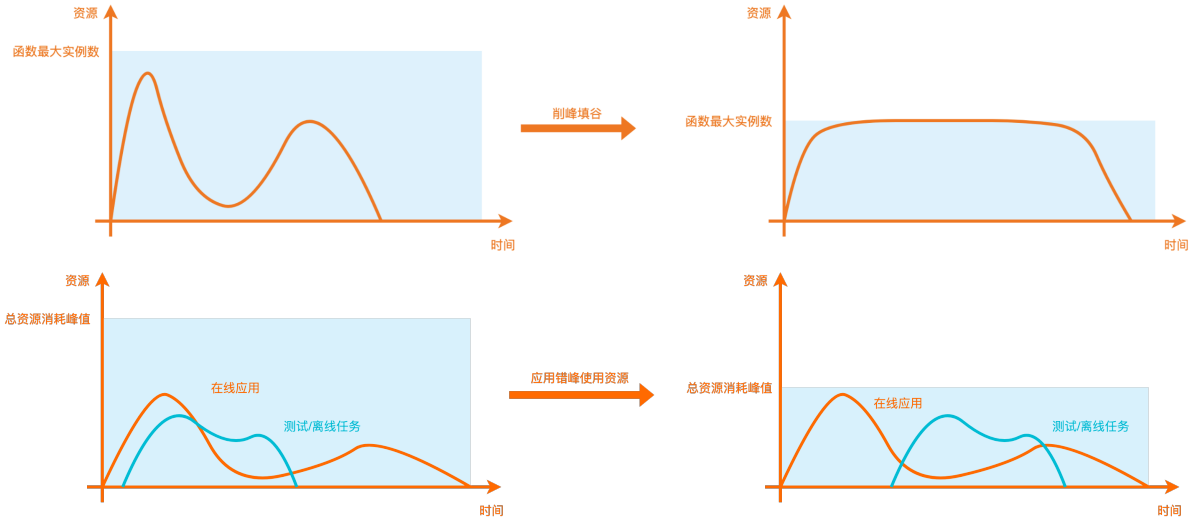
从对以上业务场景的支撑中，可以看出Serverless的表现非常优秀。与传统的应用相比，Serverless有以下差异化价值：

- 简单提效
  传统BFF（Back-end For Front-end）层应用会随着时间推移，以及业务需求的增加， 其BFF层逐渐变 “富”， 冗余的代码逐渐变多。随着人员迭代变化，模块的开发者也会变化，BFF层就会慢慢变成一个无人知晓，无人敢动的模块。

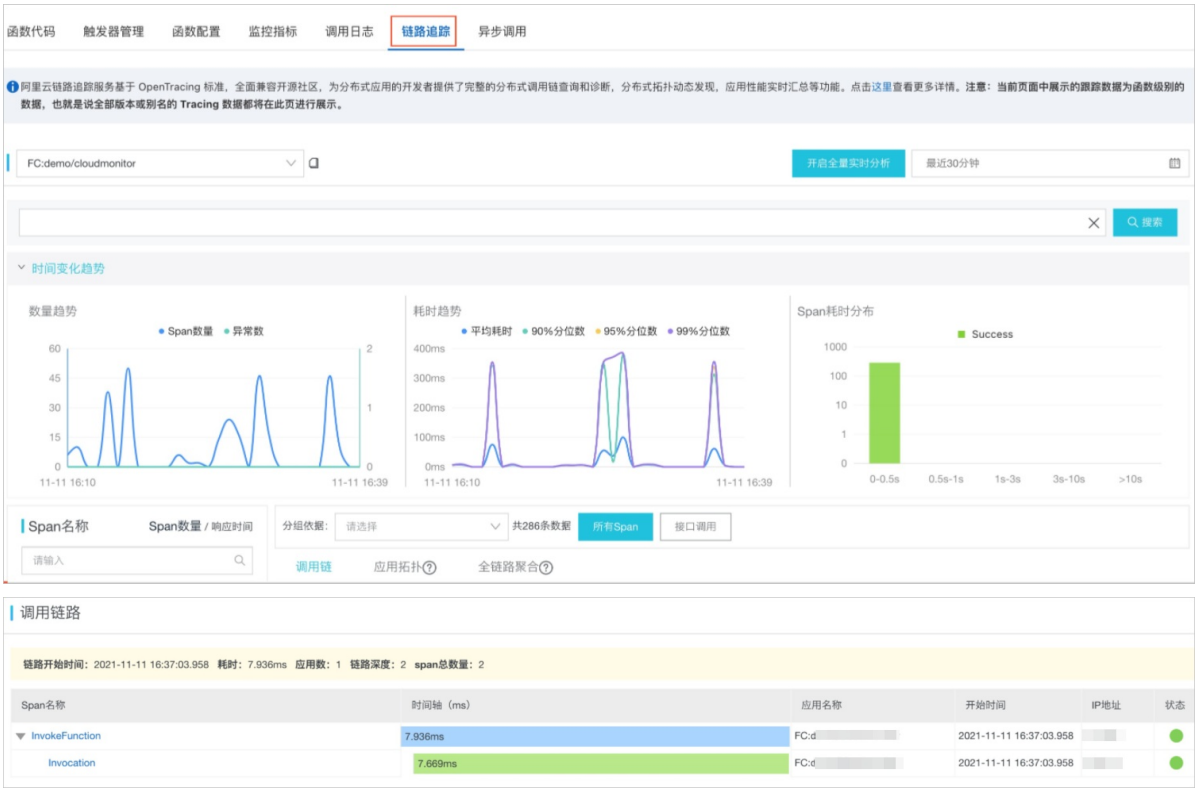
当BFF层转换成SFF (Serverless For Front-end) 层之后，SFF的职责会变的单一、零运维、成本更低，这些是Serverless本身自带的能力，而这些能力可以帮助前端进一步释放生产潜能。开发者不再需要一个富BFF层，而只需一个接口或一个SFF就可以实现功能，天然解决了牵一发而动全身的问题。如果接口停服或者没有流量，那么所用的实例会自动缩零，也就很容易分辨出是哪一个接口函数，后期就可以删掉此接口的函数，有效提升资源利用率。
高德在Serverless应用上非常先进，实现了FaaS层与研发体系的完全对接，因此，应用从开发、测试、灰度、上线的全生命周期，到具备流控、弹性、容灾等标准化能力，所用的时间较以前缩短了40%，大大提高了效率。



- 弹性以及成本
  通过流量趋势数据，高德观察到地图场景流量特点——高峰与低峰的落差十分明显。按照传统应用的资源准备，高德需要根据最高峰的流量进行资源准备，所以到了流量低峰期，多准备的机器会有很多冗余，这就造成了成本的浪费。
  针对以上情况，高德使用了阿里云函数计算，可以根据流量变化自动扩缩容。然而，提升扩缩容速度的复杂性较大，一直是大企业的专属，但函数计算可以通过毫秒级别的启动优势，将快上快下的扩缩容能力普及给用户，轻松帮助用户实现了计算资源的弹性利用，并且大大降低了成本。



- 可观测性
  可观测性是应用上线诊断平台的必备属性，要让用户观察到RT变化、资源的使用率、系统应用的全链路调用，从而快速诊断出系统应用的瓶颈问题。阿里云函数计算率先与日志服务、云监控、链路追踪平台以及函数工作流编排做了完美的融合。您只需要配置一次，就可以完整的享受到以上这些功能，大大降低了您的学习成本，实现了对应用程序的快速诊断。



Serverless规模化落地的序幕已经拉开， 更多场景正在各行各业中解锁。Serverless在高德的规模化落地，对于业务方来而言，业务迭代更快更灵活了，为业务创新创造了前提条件；对于前端开发者而言，进一步激活了开发者的生产潜能，提升了极大的能力自信。高德出行业务从2020年初的能力试点到十一出行节的自主出行核心场景，期间接入了阿里云函数计算，积累了非常宝贵的云原生落地经验，为未来业务整体上云打下了良好基础。

2.9. 拉勾教育

本文以拉勾教育为例，介绍通过函数计算、对象存储、API网关、日志服务等云服务如何应对流量洪峰等问题。

客户介绍

拉勾网隶属于北京拉勾网络技术有限公司，是一个专注于互联网职业的招聘网站，以众多优质互联网资源为依托，发布圈内招聘信息，为求职者提供人性化、个性化及专业化的信息服务，以让优质人才和优秀企业及时相遇为己任。拉勾网是一个热爱互联网的年轻团队，他们致力于打造最专业的互联网招聘平台。

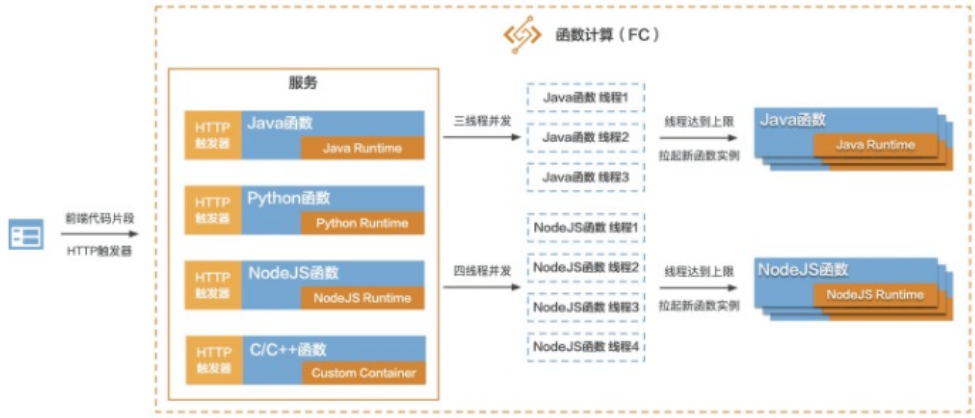
拉勾教育是北京拉勾网络技术有限公司旗下品牌，帮助互联网职场人交流和提升技术能力，是互联网人的“实战大学”。拉勾教育基于拉勾网在互联网人才服务行业的经验，共同研发出一系列课程和服务，为互联网人提供学习机会，拉勾教育已开设专栏和训练营两种学习方式，课程涵盖研发、产品、设计、运营及市场等领域。

客户痛点

在拉勾教育众多的课程中，有一块领域是编程教育，会以图、文、音、视频的方式教授互联网热门领域的编程课程，同时课程中最大特色的就是交互式学习，做到了对知识的接收、理解、动手实操及掌握的闭环，编程教育的核心是在线编程系统，即在线输入各种编程语言的代码就能实时返回执行结果。用户期望在线编程系统能够具备以下特征：

1. 支持多种编程语言Runtime：拉勾教育编程课程涉及到的编程语言、技术框架比较多，需要一套系统支持多种编程语言Runtime，并且能够方便快捷的增加其他编程语言Runtime。
2. 可以支撑大并发量的请求：拉勾教育有大量优质的师生资源，每门课都有数以万计的同学在学习，需要一套在线编程系统能够支撑To C级别的大并发请求。
3. 安全性高：不期望因为学习者的错误代码，例如死循环、黑客的恶意代码，造成系统崩溃甚至影响其他业务。

解决方案



函数计算是阿里云提供的Serverless计算平台，完美的契合了拉勾教育的诸多痛点，只需编写执行不同编程语言的逻辑并上传代码即可。以HTTP或事件驱动的方式触发函数计算实例执行业务逻辑并响应用户请求。同时与阿里云的对象存储、API网关、日志服务、MNS消息队列和表格存储等服务无缝对接，帮助用户快速构建解决流量洪峰的系统。

使用效果

1. 支持多种编程语言的Runtime：函数计算支持Java、Python、Node.js、PHP及.NET Core语言，并且支持通过Custom Container的方式来构建其他语言的Runtime，例如Go Custom Runtime、Ruby Custom Runtime、PowerShell Custom Runtime等。
2. 极致弹性和高可用性：用传统服务器搭建的系统若要支撑大并发，只能通过增加机器数量来实现，但是在流量突发场景，或者有一定流量潮汐的场景，机器启动的速度是比较慢的。函数计算通过预留实例模式和按量模式消除冷启动，达到百毫秒级拉起实例，同时可以配置单实例多并发模式，在突发流量场景下可快速拉起实例，横向扩容系统，从容面对流量洪峰。
3. 完善的安全保障机制：函数计算在产品和系统设计的每个方面都把安全作为最高优先级的目标。您的程序在函数计算环境中运行的隔离保护和阿里云的云服务器的安全级别是一样的，函数计算在虚拟机级别隔离运行，在网络、数据、代码及防DDos攻击等方面做了完备的保护。

3.权限管理

3.1. 配置权限助手

权限助手简化了函数计算相关的RAM权限策略配置，帮助您快速自定义权限和划分角色。本文介绍如何在函数计算控制台快速创建RAM权限策略草稿并在RAM控制台完成最终权限配置。

背景信息

权限助手是一个生成RAM权限策略的工具，能够协助您对函数计算的权限进行可视化配置，并在函数计算控制台生成对应的RAM权限语句。然后，您可以在RAM控制台创建对应的自定义权限策略，将权限策略的策略内容修改为函数计算控制台生成的RAM权限语句，并为需要该项权限的RAM用户授予权限。

注意事项

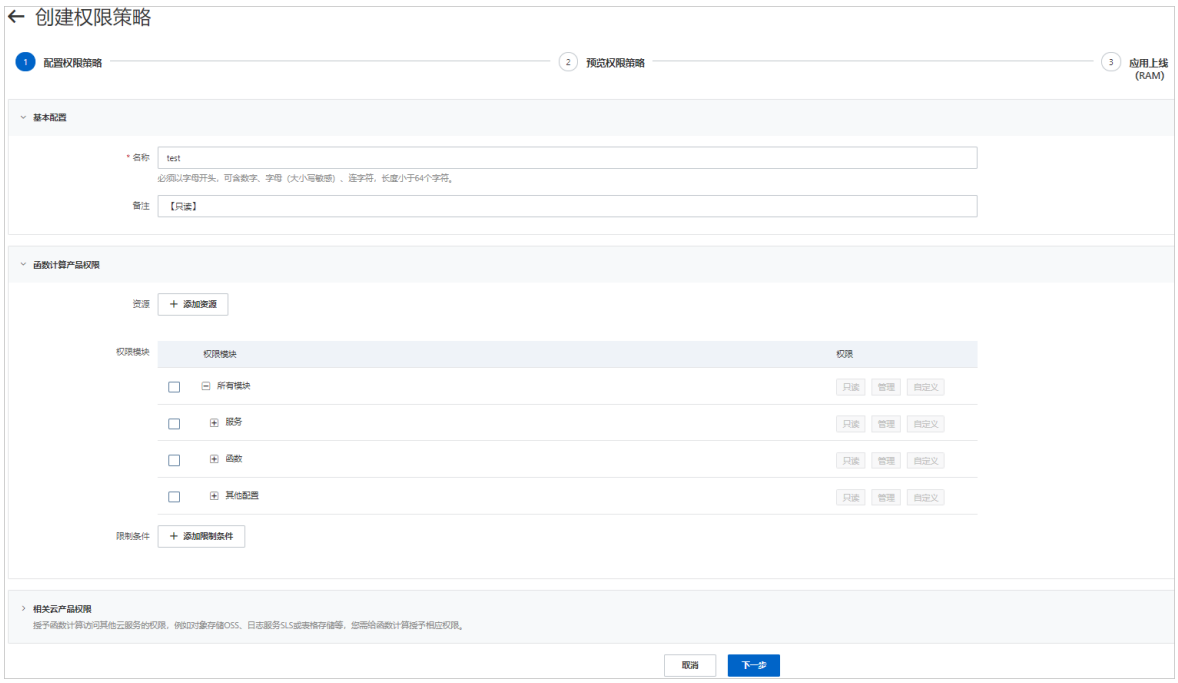
- 当函数计算发布新功能时，您需要重新生成RAM权限语句，否则可能会导致原有权限策略下的RAM用户不具备该新功能的权限。
- 权限助手功能仅用于函数计算的服务、函数、层、域名等粒度相关的权限策略配置，如您想要为账号授予更多其他产品或更细粒度的权限，请参见创建自定义权限策略。

前提条件

- 访问控制
  - 创建RAM用户
- 函数计算
  - 创建服务
  - 创建函数

步骤一：在函数计算控制台创建权限策略

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**更多功能 > 权限助手**。
3. 在**权限助手**页面，单击**创建权限策略**。
4. 在**创建权限策略**页面的**配置权限策略**配置向导页面，设置相关参数并单击**下一步**。



i. 在**基本配置**区域，配置以下参数。

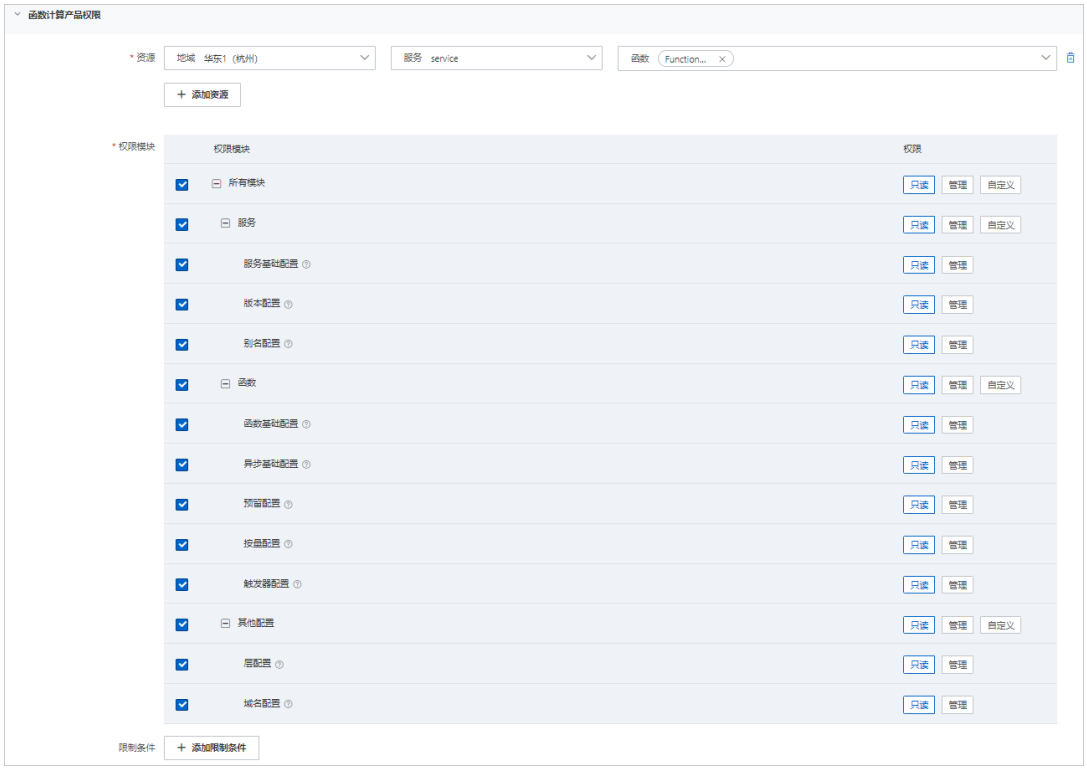
参数	描述
名称	自定义权限策略的名称。
（可选）备注	自定义权限策略的备注说明。

ii. （可选）在**函数计算产品权限**区域，按需配置以下内容。

a. 单击**+添加资源**，在下拉列表中依次选择**地域**、**服务**和**函数**。

> **说明** 如果需要在所有地域下创建权限策略，**地域**下拉列表请选择所有地域。

b. 在**权限模块**的**权限模块**和**权限**列，选中目标模块复选框及其对应权限等级。

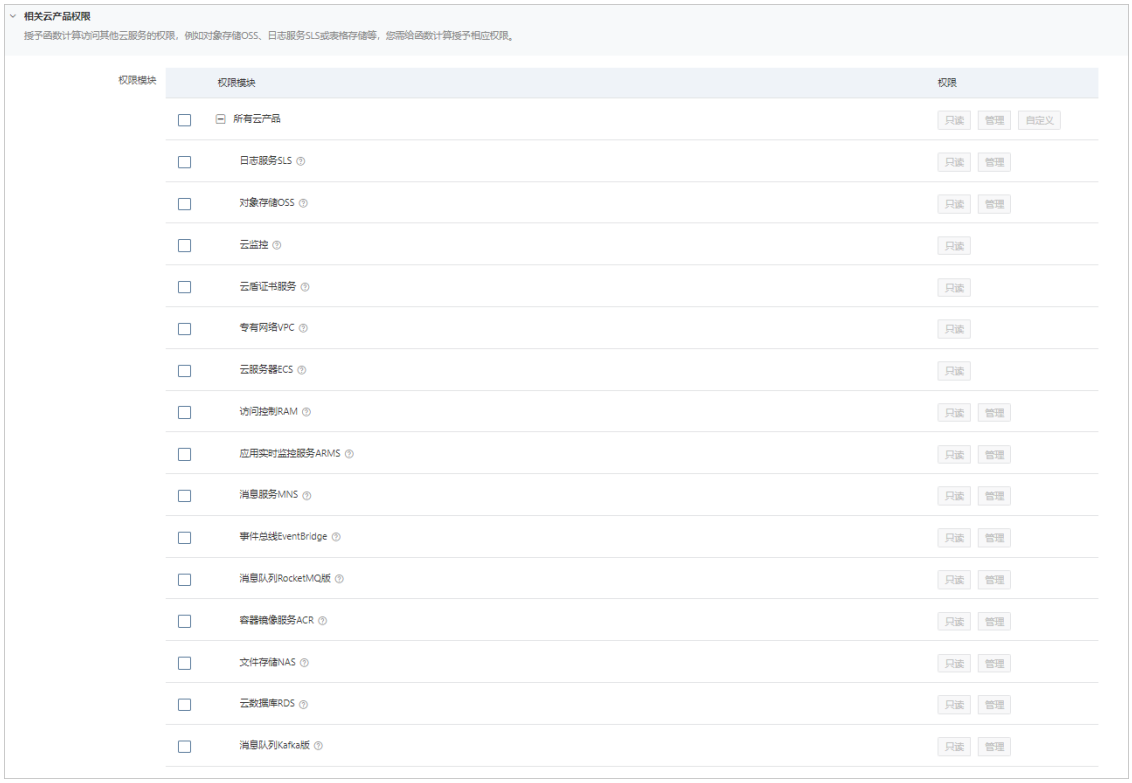


> **说明** 支持按照服务、函数、层、域名等粒度对函数计算的权限进行层级划分。

c. （可选）单击**+添加限制条件**，在下拉列表中依次选择限制条件**关键字**、**限定词**，然后输入限制条件的**值**。

关键字	限定词	值
请求时间	■ DataEquals ■ DataNotEquals ■ DataLessThan ■ DataLessThanEquals ■ DataGreaterThan ■ DataGreaterThanEquals	发送请求时间。格式为ISO 8601，例如：2021-11-11T23:59:59Z。当选择一个**限定词**时，系统默认填入当前时间。
安全信道	Bool	发送请求是否使用了安全信道。例如，HTTPS。 ■ *true*：使用。 ■ *false*：未使用。
客户端 IP	■ IpAddress ■ NotIpAddress	客户端IP地址。例如，10.0.XX.XX。
多因素认证	Bool	您登录时是否使用了多因素认证，多因素认证是指使用两种以上的认证方式进行登录。 ■ *true*：使用。 ■ *false*：未使用。

iii. （可选）在**相关云产品权限**区域，在**权限模块**和**权限**列，选择目标云产品复选框及其对应权限等级。



> **说明** 如您需要给函数计算授予访问更多其他阿里云服务的权限，请参见授予函数计算访问其他云服务的权限。

5. 在**创建权限策略**页面的**预览权限策略**配置向导页面，检查生成的规则列表，然后单击**下一步**。
   您可以在**权限策略**区域，对生成的RAM权限策略进行**压缩**、**格式化**或**复制**。复制的RAM授权语句将用于步骤二：在RAM控制台创建自定义权限策略。
6. 在**创建权限策略**页面的**应用上线（RAM）**配置向导页面，阅读在RAM控制台相关操作指引，然后单击**完成**。

步骤二：在RAM控制台创建自定义权限策略

在RAM控制台创建自定义权限策略，权限策略的内容需修改为步骤一生成的RAM权限策略。

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**权限管理 > 权限策略**。
3. 在**权限策略**页面，单击**创建权限策略**。
4. 在**创建权限策略**页面，单击**脚本编辑**页签。
5. 输入权限策略内容，然后单击**下一步：编辑基本信息**。
   关于权限策略语法结构的详情，请参见权限策略语法和结构。
6. 输入权限策略**名称**和**备注**。
7. 检查并优化权限策略内容。
   - 基础权限策略优化
     系统会对您添加的权限策略语句自动进行基础优化。基础权限策略优化会完成以下任务：
     - 删除不必要的条件。
     - 删除不必要的数组。
   - （可选）高级权限策略优化
     您可以将鼠标悬浮在**可选：高级策略优化**上，单击**执行**，对权限策略内容进行高级优化。高级权限策略优化功能会完成以下任务：

- 拆分不兼容操作的资源或条件。
- 收缩资源到更小范围。
- 去重或合并语句。

8. 单击**确定**。

步骤三：在RAM控制台为RAM用户添加授权

增加权限策略后，您需要在RAM控制台为需要该项权限的RAM用户授权。本文以在授权页面为RAM用户授权的方式为例，操作步骤如下所示。更多方式，请参见为RAM用户授权。

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**权限管理 > 授权**。
3. 在**授权**页面，单击**新增授权**。
4. 在**新增授权**页面，为RAM用户添加权限。
   i. 选择授权应用范围。
      - **整个云账号**：权限在当前阿里云账号内生效。
      - **指定资源组**：权限在指定的资源组内生效。

      > **说明** 指定资源组授权生效的前提是该云服务已支持资源组。更多信息，请参见支持资源组的云服务。

   ii. 输入授权主体。

      授权主体即需要授权的RAM用户。

   iii. 选择权限策略。

      > **说明** 每次最多绑定5条策略，如需绑定更多策略，请分次操作。

5. 单击**确定**。
6. 单击**完成**。

更多信息

除了函数计算控制台，您还可以通过API或SDK配置权限助手。更多信息，请参见SDK列表。

3.2. 授予函数计算访问其他云服务的权限

当函数计算需访问阿里云其他云服务时例如对象存储OSS、日志服务SLS或表格存储等，您需给函数计算授予相应权限。该权限是服务级别的，当服务被授予某权限后该服务下所有函数都将具有此权限。本文以授予函数计算访问对象存储权限为例，介绍如何授予函数计算访问其他云服务的权限。

前提条件

创建服务

默认服务角色

在函数运行的过程中，函数计算需要访问其他云资源，例如将函数日志写入到您指定的日志服务内、拉取ACR镜像或打通VPC网络访问等。为了简化您的授权操作，函数计算为您提供了一个系统默认的服务角色即AliyunFCDefaultRole。该角色内包含了函数计算需要访问的部分云资源权限，关于如何绑定该角色的操作步骤，请参见开通服务。该角色的授权内容如下所示：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "vpc:DescribeVSwitchAttributes"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ecs:CreateNetworkInterface",
        "ecs:DeleteNetworkInterface",
        "ecs:DescribeNetworkInterfaces",
        "ecs:CreateNetworkInterfacePermission",
        "ecs:DescribeNetworkInterfacePermissions",
        "ecs:DeleteNetworkInterfacePermission"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:PostLogStoreLogs"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "cr:GetRepository",
        "cr:GetRepositoryTag",
        "cr:GetAuthorizationToken",
        "cr:PullRepository"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "fc:InvokeFunction",
        "mns:SendMessage",
        "mns:PublishMessage",
        "eventbridge:PutEvents",
        "mq:PUB",
        "mq:OnsInstanceBaseInfo"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

由于AliyunFCDefaultRole角色包含的权限是粗粒度的，如果需要更细粒度的权限，您也可以为服务授予其他角色及相关权限策略。

例如您想让函数计算某服务内的所有函数都具有管理对象存储服务的权限，但AliyunFCDefaultRole角色内没有该权限。您需在配置服务权限时，为该服务绑定一个角色，然后将管理对象存储服务的权限策略绑定到该角色上。成功绑定后，该服务内的所有函数都将具有管理对象存储服务的权限了。详细操作，请参见操作步骤。

> **注意** 由于AliyunFCDefaultRole是所有服务级别的角色，如果您在添加默认角色的基础上还想添加其他权限策略，建议新增一个角色，然后给该角色添加相关权限，请勿在默认角色AliyunFCDefaultRole上添加其他权限策略。

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在编辑服务页面的**角色配置**区域，配置相关参数，然后单击**保存**。
   - 使用新角色
     a. 单击**创建新的服务角色**，页面跳转至**角色**页面。
     b. 在**角色**页面，创建可信实体为阿里云服务的RAM角色。具体操作步骤，请参见创建可信实体为阿里云服务的RAM角色。

        > **说明** 本示例中目标授信服务选择**函数计算**。

     c. 给新角色授予权限策略。您可以选择添加新策略或者使用已有策略。具体操作步骤，请分别参见创建自定义权限策略和为RAM角色授权。
   - 使用已有角色
     在**角色配置**区域，从**服务角色**下拉列表中选择需要授权的角色。

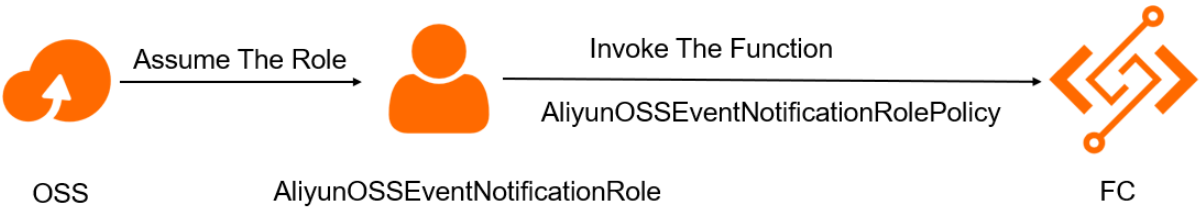
     > **说明** 如果您选择的服务角色不具有管理对象存储服务的权限，您需要给该角色添加此权限。具体操作步骤，请参见为RAM角色授权。

3.3. 授予事件源访问函数计算的权限

当您需通过事件源如对象存储OSS触发函数执行时，可授予该事件源触发函数的权限，该权限是触发器级别的，每个触发器的角色均可以被授予相应权限，该角色也可以同时被授予不同触发器。本文以授予OSS访问函数计算的权限为例，介绍授予事件源访问函数计算权限的应用场景和操作步骤。

应用场景

以OSS事件源触发函数计算代码执行为例，当存在一个OSS Bucket A和函数B，将上传文件到Bucket A的过程，作为触发函数计算函数B的事件时，您需要创建一个OSS触发器并授予相应权限策略。访问控制提供系统授权策略AliyunOSSEventNotificationRole，该策略具有OSS事件源触发函数计算代码执行的权限。在创建触发器时，您可以为触发器绑定一个角色，既可以新建一个新角色也可以使用已有角色。然后将策略AliyunOSSEventNotificationRole绑定到该角色内，成功绑定该策略后OSS事件源就可以触发函数计算代码执行。



操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务**操作**列的**函数管理**。
5. 在**函数管理**页面，单击目标函数名称。

6. 在函数详情页面，单击**触发器管理**页签，从**版本或别名**下拉列表选择要创建触发器的版本或别名，然后单击**创建触发器**。
7. 在创建触发器面板，填写相关信息。然后单击**确定**。

> **说明** 其他类型的触发器的权限配置过程基本一致，按需选择相应的服务类型即可。HTTP触发器事件源在您创建HTTP函数时已成功创建。

参数	操作	本文示例
触发器类型	选择**对象存储 OSS**。	对象存储 OSS
名称	填写自定义的触发器名称。	oss-trigger
版本或别名	默认值为**LATEST**，如果您需要创建其他版本或别名的触发器，需先在函数详情页的**版本或别名**下拉列表选择该版本。关于版本和别名的简介，请参见管理版本和管理别名。	LATEST
Bucket 名称	选择已创建的OSS Bucket。	bucket-zh****
文件前缀	输入要匹配的文件名称的前缀。强烈建议您配置前缀和后缀，避免触发事件嵌套循环触发引起额外费用。另外，一个Bucket的不同触发器如果指定了相同的事件类型，则前缀和后缀不能重复。详细信息，请参见触发规则。 **注意** 文件前缀不能以 `/` 开头。	source
文件后缀	输入要匹配的文件名称的后缀。强烈建议您配置前缀和后缀，避免触发事件嵌套循环触发引起额外费用。另外，一个Bucket的不同触发器如果指定了相同的事件类型，则前缀和后缀不能重复。详细信息，请参见触发规则。	zip
触发事件	选择一个或多个触发事件。关于对象存储OSS的事件类型，请参见OSS事件定义。 本示例选择**oss:ObjectCreated:PutObject**。	oss:ObjectCreated:PutObject
角色名称	选择**AliyunOSSEventNotificationRole**。 **说明** 如果您第一次创建该类型的触发器，则需要在单击**确定**后，在弹出的对话框中选择**立即授权**。	AliyunOSSEventNotificationRole

8. 单击**确定**。

3.4. 通过阿里云账号给RAM用户授权

借助访问控制RAM的RAM用户，您可以实现阿里云账号和RAM用户权限分割，按需为RAM用户赋予不同的权限，仅限有权限的RAM用户在函数计算控制台上管理资源，并避免因暴露阿里云账号密钥而造成安全风险。本文介绍如何通过阿里云账号创建RAM用户，并为此RAM用户授权。

应用场景

企业A开通了函数计算服务，该企业需要员工操作函数计算的相关资源，例如创建服务、删除服务、创建函数及删除函数等。由于每个员工的工作职责不一样，所以需要的权限也不一样。企业A的需求如下：

- 出于安全或信任的考虑，不希望将云账号密钥直接透露给员工，而希望给员工创建相应的用户账号。
- 用户账号只能在授权的前提下操作资源，不需要对用户账号进行独立的计量计费，所有开销都计入企业账号名下。
- 随时可以撤销用户账号的权限，也可以随时删除其创建的用户账号。

步骤一：使用企业A的阿里云账号为员工创建RAM用户

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 用户**。
3. 在**用户**页面，单击**创建用户**。
4. 在**创建用户**页面的**用户账号信息**区域，输入**登录名称**和**显示名称**。

   > **说明** 单击**添加用户**，可一次性创建多个RAM用户。

5. 在**访问方式**区域，选择访问方式。
   - **控制台访问**：设置控制台登录密码、重置密码策略和多因素认证策略。

     > **说明** 自定义登录密码时，密码必须满足密码复杂度规则。关于如何设置密码复杂度规则，请参见设置RAM用户密码强度。

   - **OpenAPI调用访问**：自动为RAM用户生成访问密钥（AccessKey），支持通过API或其他开发工具访问阿里云。

     > **说明** 为了保障账号安全，建议仅为RAM用户选择一种登录方式，避免RAM用户离开组织后仍可以通过访问密钥访问阿里云资源。

6. 单击**确定**。

步骤二：为RAM用户授权

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 用户**。
3. 在**用户**页面，单击目标RAM用户**操作**列的**添加权限**。
4. 在**添加权限**面板，为RAM用户添加权限。
   i. 选择授权应用范围。
      - **整个云账号**：权限在当前阿里云账号内生效。
      - **指定资源组**：权限在指定的资源组内生效。

        > **说明** 指定资源组授权生效的前提是该云服务已支持资源组。更多信息，请参见支持资源组的云服务。

   ii. 输入授权主体。

   授权主体即需要授权的RAM用户，系统会自动填入当前的RAM用户，您也可以添加其他RAM用户。

   iii. 选择权限策略。

   > **说明** 每次最多绑定5条策略，如需绑定更多策略，请分次操作。

5. 单击**确定**。
6. 单击**完成**。

> **说明** 以上为RAM用户授权操作中，选择的权限策略可以为系统策略，也可以为自定义策略。具体信息，请参见权限策略及示例。您可以自主创建、更新和删除自定义策略。具体操作步骤，请参见创建自定义权限策略。

后续步骤

使用阿里云账号创建好RAM用户后，企业A即可将创建好的RAM用户的登录名称及密码或AccessKey信息分发给使用者。使用者可以按照以下步骤使用RAM用户登录控制台或调用API。

- 控制台

i. 在浏览器打开RAM用户登录入口。

ii. 在**RAM用户登录**页面，输入RAM用户名称，单击**下一步**，输入RAM用户密码，然后单击**登录**。

> **说明** RAM用户登录名称的格式为*<$username>@<$AccountAlias>*或*<$username>@<$AccountAlias>.onaliyun.com*。*<$AccountAlias>*为账号别名，如果没有设置账号别名，则默认值为阿里云账号的ID。

- API
  使用RAM用户的AccessKey调用API，您可以在代码中使用RAM用户的AccessKey ID和AccessKey Secret调用API访问函数计算。

3.5. 通过RAM角色实现跨云账号授权

本文介绍如何使用RAM控制台和SDK获取阿里云临时安全令牌STS（Security Token Service）并实现跨账号授权查看函数计算的资源的操作步骤。

使用示例

企业A开通了函数计算，该企业需要企业B代为操作函数计算的资源。同时企业A有如下诉求：

- 企业A希望能专注于业务系统，仅作为函数计算的所有者，同时，可以授权企业B操作部分业务，例如创建服务、创建函数等。
- 企业A希望当企业B的员工加入或离职时，无需做任何权限变更。企业B可以进一步将企业A的资源访问权限分配给企业B的RAM用户，并可以精细控制其RAM用户对资源的访问或操作权限。
- 企业A希望如果双方合同终止，企业A随时可以撤销对企业B的授权。

使用控制台的操作步骤

假设企业A需要授权企业B的员工访问函数计算的所有服务。企业A和企业B下分别有一个阿里云账号A和阿里云账号B：

- 企业A的阿里云账号ID为`123456789012****`，账号别名（企业别名）为`company-a`。
- 企业B的阿里云账号ID为`134567890123****`，账号别名（企业别名）为`company-b`。

使用阿里云账号A创建一个RAM角色，并为RAM角色授予适当的权限，允许阿里云账号B使用该角色。

1. 使用阿里云账号A登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 角色**。
3. 在**角色**页面，单击**创建角色**。
4. 在**创建角色**面板的**选择类型**配置向导页面，单击**阿里云账号**，然后单击**下一步**。
5. 在**创建角色**面板的**配置角色**配置向导页面，设置角色名称例如fc-admin，单击**其他云账号**并填写企业B的阿里云账号。
6. 单击**完成**。
   如果页面跳转至**创建角色**面板的**创建完成**配置向导页面，表示创建RAM角色成功。RAM角色创建成功后，您可以在该角色的基本信息页面内查看到该RAM角色的ARN和信任策略：
   - RAM角色的ARN为：`acs:ram::123456789012****:role/fc-admin`。
   - RAM角色的信任策略如下：

     > **说明** 以下策略表示仅允许阿里云账号B下的RAM用户来扮演该RAM角色。

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "RAM": [
          "acs:ram::134567890123****:root"
        ]
      }
    }
  ],
  "Version": "1"
}
```

7. 阿里云账号A为RAM角色fc-admin添加AliyunFCReadOnlyAccess权限。关于如何为RAM角色授权的操作步骤，请参见为RAM角色授权。

1. 使用阿里云账号B为其员工创建RAM用户test-demo。关于如何创建RAM用户的操作步骤，请参见创建RAM用户。
2. 阿里云账号B为创建好的RAM用户添加AliyunSTSAssumeRoleAccess权限，即允许RAM用户扮演RAM角色。关于如何为RAM用户添加权限，请参见为RAM用户授权。

当阿里云账号B下的某个员工（RAM用户）需要访问阿里云账号A下的资源时，阿里云账号B可以自主进行授权控制。即阿里云账号B下的RAM用户扮演阿里云账号A下的RAM角色访问阿里云账号A下的资源。具体操作如下：

1. 使用阿里云账号B的RAM用户登录RAM控制台。关于RAM用户登录控制台的详细信息，请参见RAM用户登录阿里云控制台。
2. 成功登录后，您需要将鼠标悬停在右上角头像的位置，单击**切换身份**。
3. 在**阿里云角色切换**页面，输入相关参数信息。

   关于切换账号的详细信息，请参见使用RAM角色。

如果企业A与企业B的合作终止了，企业A只需要撤销阿里云账号B对RAM角色的使用即可。此时阿里云账号B下的所有RAM用户对RAM角色的使用权限将被自动撤销。具体操作如下：

1. 阿里云账号A登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 角色**。
3. 选择目标RAM角色即fc-admin，在**操作**列单击**删除**，然后单击**确定**。

> **注意** 在删除RAM角色前，请先为RAM角色移除权限。详情信息，请参见为RAM角色移除权限。

步骤一：阿里云账号A创建RAM角色

步骤二：阿里云账号B创建一个RAM用户

步骤三：切换身份登录

（可选）撤消授权

使用SDK的操作步骤

函数计算可以通过STS进行临时授权访问。STS是为云计算使用者提供临时访问令牌的Web服务。假设有阿里云账号A和阿里云账号B，阿里云账号B想获取查看阿里云账号A下函数计算所有服务的权限。

1. 通过以下代码示例获取访问凭证。更多信息，请参见STS SDK概览和AssumeRole。

   预期输出。

```
{
  "RequestId": "964E0EC5-575B-4FF5-8FD0-D4BD8025602A",
  "AssumedRoleUser": {
    "Arn": "acs:ram::****:role/wss/wss",
    "AssumedRoleId": "***********:wss"
  },
  "Credentials": {
    "SecurityToken": "*************",
    "AccessKeyId": "STS.*************",
    "AccessKeySecret": "*************",
    "Expiration": "2021-05-28T11:23:19Z"
  }
}
```

说明 关于在获取 `SecurityToken` 时可能遇到的常见问题，请参见RAM角色和STS Token常见问题。

Node.js Python

2. 修改阿里云账号B的函数代码如下所示，使其RAM用户有权限查看阿里云账号A下函数计算的所有服务。

```
const FC = require('@alicloud/fc2');
// 构建客户端。
const client = new FC(<accountID>, {
    region: <yourRegionID>,
    accessKeyID: <yourAccessKeyID>,
    securityToken: <yourSecurityToken>,
    accessKeySecret: <yourAccessKeySecret>
});
// 获取服务列表。
client.listServices().then(res => {
    console.log(JSON.stringify(res, null, ' '))
}).catch(ex=> console.log(ex))
```

注意 临时密钥的授权角色需要有获取服务列表的权限。

3.6. 权限策略及示例

函数计算权限管理是通过阿里云的访问控制RAM实现的。使用访问控制RAM可以让您避免与其他用户共享云账号密钥，即AccessKey（包含AccessKey ID和AccessKey Secret），按需为RAM用户分配最小权限。本文介绍函数计算的权限策略，包括系统策略、自定义策略及自定义策略示例。

策略类型

在访问控制中，权限策略是用语法和结构描述的一组权限的集合，可以精确地描述被授权的Resource（资源集）、Action（操作集）以及授权条件。函数计算包含以下权限策略：

- 系统策略：统一由阿里云创建，您只能使用不能修改，策略的版本更新由阿里云维护。
- 自定义策略：您可以自主创建、更新和删除，策略的版本更新由您自己维护。

系统策略

当您首次使用RAM用户登录函数计算控制台时，不仅需要通过阿里云账号给您的RAM用户添加访问函数计算的系统权限策略，也需要给RAM用户添加访问其他云服务的系统权限策略。成功授权后，您的RAM用户才可以正常访问函数计算服务及其他云产品服务。

系统策略包含以下类型：

- 函数计算服务提供的系统策略。

权限策略名称	描述
AliyunFCReadOnlyAccess	表示允许对函数计算所有的资源进行读操作。
AliyunFCInvocationAccess	表示允许对所有函数的资源进行执行操作。
AliyunFCFullAccess	表示允许对所有函数计算资源进行所有执行操作。

> **说明** 管理函数计算服务FC权限AliyunFCFullAccess内包含调用函数计算FC服务函数的权限AliyunFCInvocationAccess及只读访问函数计算FC服务权限AliyunFCReadOnlyAccess。当成功授权管理函数计算服务的权限后，您无需再授权调用或只读函数计算的权限。

- 其他云产品服务提供的系统策略。

云产品名称	系统策略
日志服务SLS	○ AliyunLogReadOnlyAccess：只读访问日志服务Log的权限。 ○ AliyunLogFullAccess：管理日志服务Log的权限。 > **说明** 基于最小权限的原则，您只需授予RAM用户只读访问日志服务Log的权限AliyunLogReadOnlyAccess就可以访问日志服务Log了。
对象存储OSS	○ AliyunOSSReadOnlyAccess：只读对象存储服务OSS的权限。 ○ AliyunOSSFullAccess：管理对象存储服务OSS权限。
云监控	AliyunCloudMonitorReadOnlyAccess：只读访问云监控CloudMonitor的权限。
云盾证书服务	AliyunYundunCertReadOnlyAccess：只读访问云盾证书服务的权限。
专有网络VPC	AliyunVPCReadOnlyAccess：只读访问专有网络VPC的权限。
云服务器ECS	AliyunECSReadOnlyAccess：只读访问云服务器服务ECS的权限。
访问控制RAM	○ AliyunRAMReadOnlyAccess：只读访问控制RAM的权限，即查看用户、组以及授权信息的权限。 ○ AliyunRAMFullAccess：管理访问控制RAM的权限，即管理用户以及授权的权限。 > **说明** AliyunRAMReadOnlyAccess仅适用于在控制台获取角色的列表，当RAM用户还需要进行其他操作，您需给RAM用户添加管理访问控制RAM的权限即AliyunRAMFullAccess。
应用实时监控服务ARMS	○ AliyunARMSReadOnlyAccess：只读访问业务实时监控服务ARMS的权限。 ○ AliyunARMSFullAccess：管理业务实时监控服务ARMS的权限。 > **说明** 基于最小权限的原则，您只需授予RAM用户只读访问业务实时监控服务ARMS的权限AliyunARMSReadOnlyAccess就可以访问应用实时监控服务ARMS了。

云产品名称	系统策略
消息服务MNS	◦ AliyunMNSReadOnlyAccess：只读访问消息服务MNS的权限。 ◦ AliyunMNSFullAccess：管理消息服务MNS的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户只读访问消息服务MNS的权限即AliyunMNSReadOnlyAccess就可以满足访问消息服务MNS了。
事件总线EventBridge	◦ AliyunEventBridgeReadOnlyAccess：只读访问事件总线EventBridge的权限。 ◦ AliyunEventBridgeFullAccess：管理事件总线EventBridge的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户只读访问事件总线EventBridge的权限即AliyunEventBridgeReadOnlyAccess就可以满足访问事件总线EventBridge了。
消息队列RocketMQ版	◦ AliyunMQReadOnlyAccess：只读访问消息队列MQ的权限。 ◦ AliyunMQFullAccess：管理消息队列MQ的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户只读访问消息队列MQ的权限即AliyunMQReadOnlyAccess就可以满足访问消息队列RocketMQ版了。
容器镜像服务ACR	◦ AliyunContainerRegistryReadOnlyAccess：只读访问容器镜像服务ContainerRegistry的权限。 ◦ AliyunContainerRegistryFullAccess：管理容器镜像服务ContainerRegistry的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户只读访问容器镜像服务ContainerRegistry的权限即AliyunContainerRegistryReadOnlyAccess就可以满足访问容器镜像服务了。
文件存储NAS	◦ AliyunNASReadOnlyAccess：查看文件存储服务NAS的权限。 ◦ AliyunNASFullAccess：管理文件存储服务NAS的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户查看文件存储服务（NAS）的权限即AliyunNASReadOnlyAccess就可以满足访问文件存储服务了。
云数据库RDS	◦ AliyunRDSReadOnlyAccess：只读访问云数据库服务RDS的权限。 ◦ AliyunRDSFullAccess：管理云数据库服务RDS的权限。 **说明** 基于最小权限的原则，您只需授予RAM用户只读访问云数据库服务RDS的权限即AliyunRDSReadOnlyAccess就可以满足访问云数据库服务了。
云效	AliyunRDCDevStudioRole：表示帮助您创建云效账号及初始化云效企业。

注意 当阿里云账号给RAM用户授予关于触发器相关的权限例如对象存储OSS的AliyunOSSFullAccess权限后，出现无法更新触发器的情况时，阿里云账号还需给RAM用户添加如下自定义策略，成功授权后RAM用户就可以正常更新对象存储触发器了。

```
{
        "Statement": [
            {
                "Action": [
                    "ram:PassRole"
                ],
                "Effect": "Allow",
                "Resource": "*"
            }
        ],
        "Version": "1"
    }
```

自定义策略

除了函数计算默认提供的系统策略外，您也可以通过自定义策略进行更细粒度的权限管理。关于权限策略的基本信息，请参见权限策略基本元素。

Resource	Action	描述
acs:fc:<region>:<account-id>:services/<serviceName>	fc:GetService	特定服务资源。
	fc:UpdateService	
	fc:DeleteService	
acs:fc:<region>:<account-id>:services/*	fc:CreateService	所有服务资源。
	fc:ListServices	
acs:fc:<region>:<account-id>:services/<serviceName>.<qualifier>	fc:GetService	特定版本的服务资源。
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>	fc:GetFunction	特定服务下的特定函数资源。
	fc:UpdateFunction	
	fc:DeleteFunction	
	fc:InvokeFunction	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/*	fc:CreateFunction	特定服务下的所有函数资源。
	fc:ListFunctions	
	fc:GetFunction	
	fc:UpdateFunction	
	fc:DeleteFunction	
	fc:InvokeFunction	
	fc:PutProvisionConfig	

Resource	Action	描述
acs:fc:<region>:<account-id>:services/<serviceName>.*/functions/<functionName>	fc:GetProvisionConfig	特定服务的所有版本下的所有函数资源。
	fc:PutFunctionOnDemandConfig	
	fc:DeleteFunctionOnDemandConfig	
	fc:PutFunctionAsyncInvokeConfig	
	fc:DeleteFunctionAsyncInvokeConfig	
	fc:GetFunctionAsyncInvokeConfig	
	fc:GetFunctionOnDemandConfig	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>/triggers/<triggerName>	fc:GetTrigger	特定服务下的特定函数下的特定触发器资源。
	fc:UpdateTrigger	
	fc:DeleteTrigger	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>/triggers/*	fc:CreateTrigger	特定服务下的特定函数下的所有触发器资源。
	fc:ListTriggers	
acs:fc:<region>:<account-id>:services/<serviceName>/versions	fc:PublishServiceVersion	所有版本。
	fc:ListServiceVersions	
acs:fc:<region>:<account-id>:services/<serviceName>/versions/<versionID>	fc:DeleteServiceVersion	指定版本。
acs:fc:<region>:<account-id>:services/<serviceName>/aliases/*	fc:CreateAlias	所有别名。
	fc:ListAliases	
acs:fc:<region>:<account-id>:services/<serviceName>/aliases/<aliasName>	fc:GetAlias	指定别名。
	fc:UpdateAlias	
	fc:DeleteAlias	
acs:fc:<region>:<account-id>:custom-domains/*	fc:CreateCustomDomain	所有自定义域名。
	fc:ListCustomDomains	
acs:fc:<region>:<account-id>:custom-domains/<domainName>	fc:GetCustomDomain	指定自定义域名。
	fc:UpdateCustomDomain	
	fc:DeleteCustomDomain	
acs:fc:<region>:<account-id>:tag	fc:TagResource	单个标签。
	fc:GetResourceTags	
	fc:UnTagResource	
acs:fc:<region>:<account-id>:tags/*	fc:ListTaggedResources	所有标签。

Resource	Action	描述
acs:fc:<region>:<account-id>:account-settings/*	fc:GetAccountSettings	用户设置。
acs:fc:<region>:<account-id>:layerarn/:arn	fc:GetLayerVersionByArn	所有层。
acs:fc:<region>:<account-id>:layers/*	fc:ListLayers	
acs:fc:<region>:<account-id>:layers/:layerName/versions/:version	fc:PublishLayerAsPublic	
acs:fc:<region>:<account-id>:layers/:layerName/versions/*	fc:ListLayerVersions	所有层版本。
	fc:CreateLayerVersion	
acs:fc:<region>:<account-id>:layers/:layerName/versions/:version	fc:GetLayerVersion	
	fc:DeleteLayerVersion	
acs:fc:<region>:<account-id>:on-demand-configs/*	fc:ListOnDemandConfigs	按量实例配置。
acs:fc:<region>:<account-id>:provision-configs/*	fc:ListProvisionConfigs	预留实例配置。
acs:fc:<region>:<account-id>:services/:serviceName/binding	fc:DeleteVpcBinding	VPC绑定。
acs:fc:<region>:<account-id>:services/:serviceName/binding/*	fc:CreateVpcBinding	
	fc:ListVpcBindings	
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/async-invoke-configs/*	fc:ListFunctionAsyncInvokeConfigs	异步调用配置。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/code	fc:GetFunctionCode	所有函数代码。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/stateful-async-invocations/*	fc:ListStatefulAsyncInvocations	有状态异步调用。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/stateful-async-invocations/:invocationId	fc:GetStatefulAsyncInvocation	
	fc:StopStatefulAsyncInvocation	

您可以通过以上自定义的权限策略设置具有调用华东1（杭州）地域下*test*服务下*demo*函数的权限，具体策略如下所示：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "fc:InvokeFunction"
            ],
            "Resource": "acs:fc:cn-hangzhou:*:services/test/functions/demo",
            "Effect": "Allow"
        }
    ]
}
```

权限策略示例

```
{
    "Version":"1",
    "Statement":[
        {
            "Action":[
                "fc:CreateService",
                "fc:GetService",
                "fc:CreateFunction",
                "fc:GetFunction",
                "fc:InvokeFunction"
            ],
            "Resource":"*",
            "Effect":"Allow"
        },
        {
            "Action":[
                "ram:PassRole"
            ],
            "Effect":"Allow",
            "Resource":"*"
        }
    ]
}
```

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "log:ListProject",
                "log:ListLogStore"
            ],
            "Resource": "acs:log:*:*:project/*"
        }
    ]
}
```

```
{
  "Statement": [
    {
      "Action": [
        "oss:ListBucket",
        "oss:GetBucketEventNotification",
        "oss:PutBucketEventNotification",
        "oss:DeleteBucketEventNotification"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ],
  "Version": "1"
}
```

自定义创建获取服务及创建执行函数权限策略

自定义访问日志权限策略

自定义访问OSS触发器权限策略

3.7. 权限管理FAQ

3.7.1. 如何让其他用户为我的函数授予访问其他资源的权限？

假设用户A要授权用户B的函数访问用户A的云资源，可采用以下步骤：

1. 用户A创建一个角色，授权对象是user-B@fc.aliyuncs.com，并指定Service Role。
2. 用户B调用函数，在函数中使用context.credentials去扮演步骤1中创建的角色。
3. 用户B拿到扮演角色所得到的Credentials去访问用户A的资源。

> **说明** 在您使用日志服务时，也会遇到类似的场景。例如把用户A的日志投递到用户B的OSS Bucket中。

4.服务管理

4.1. 管理服务

本文介绍函数计算的服务的定义，以及如何在函数计算控制台上创建、更新和删除服务。

什么是服务

服务是函数计算资源管理的单位。创建函数前必须先创建服务，同一个服务下的所有函数共享一些相同的设置，例如服务授权、日志配置。

从业务场景出发，一个应用可以拆分为多个服务。从资源使用维度出发，一个服务可以由多个函数组成。例如一个数据处理服务，分为数据准备和数据处理两部分。数据准备函数资源需求小，可以选择小规格实例。数据处理函数资源需求大，可以选择大规格实例。

创建服务

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击**创建服务**。



5. 在**创建服务**面板，填写服务名称和描述，按需设置日志与链路追踪功能，然后单击**确定**。



参数说明

参数	说明
日志功能	是否启用阿里云日志服务。取值说明如下： ○ **启用**：启用后，函数计算会自动为您在日志服务创建对应的Project（日志项目）和Logstore（日志仓库），并会自动把函数日志投递到相应的Logstore。此时，函数计算会为您的服务授予基于最小权限原则创建的 `AliyunFCDefaultRole` ，以访问日志服务。服务角色的更多信息，请参见授予函数计算访问其他云服务的权限。 ○ **禁用**：函数的执行日志将无法通过日志服务存储和查询。 **说明** 请确保您已开通日志服务。否则，无法启用日志功能。
链路追踪功能	是否启用阿里云链路追踪功能。取值说明如下： ○ **启用**：启用后，您可以使用Jaeger上传链路信息，使您能够跟踪函数的执行，帮助您快速分析和诊断Serverless架构下的性能瓶颈。更多信息，请参见链路追踪简介。 ○ **禁用**：不启用链路追踪功能。 **注意** 请确保您已开通链路追踪服务。否则，无法启用链路追踪功能。

创建完成后，在**服务及函数**页面的**服务列表**页面，可以查看到已创建的服务及其配置信息。

更新服务

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。

3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在**编辑服务**页面，按需设置相应配置项，然后单击**保存**。
   - （可选）**基础信息**
     配置服务的基础信息。目前仅支持修改服务描述。

参数	说明	参考文档
描述	服务的描述，帮助您辨识服务的用途。	无。

   - （可选）**其他配置**

配置项	说明	参考文档
角色配置	配置服务中函数所使用的角色，从而使函数可以获得角色所拥有的权限。	授予函数计算访问其他云服务的权限
日志配置	配置服务的日志功能，启用日志功能后，您可以查看函数的执行日志，从而方便您执行代码调试、故障分析、数据分析等操作。	■ 配置日志 ■ 基于请求级别指标创建Grafana大盘 ■ 实例级别指标
链路追踪配置	启用链路追踪功能后，您可以使用Jaeger上传链路信息，使您能够跟踪函数的执行，帮助您快速分析和诊断Serverless架构下的性能瓶颈。	配置链路追踪
网络配置	配置服务中函数使用的网络，例如配置函数是否可以访问公网、是否可以访问VPC中的资源等。	配置网络
存储配置	配置服务中函数使用的NAS文件系统。配置后您的函数可以像访问本地文件系统一样访问存储在NAS文件系统上的文件。	配置NAS文件系统

删除服务

> 注意 删除服务前，请确保您的服务中没有函数、预留的函数实例、版本或别名，否则会导致删除失败。

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，找到需删除的目标服务，单击目标服务名称。
5. 在**服务详情**页面右上角，单击**删除**。
6. 在弹出的确认删除对话框中，单击**确认**。
   在**服务列表**页面，无法查看到已删除的服务信息。

更多信息

除了函数计算控制台，您还可通过Serverless Devs工具管理服务。更多信息，请参见服务相关命令。

4.2. 管理版本

函数计算支持版本管理功能，帮助您更高效地管理服务、函数以及触发器。您可以通过版本管理功能发布多个版本的服务，实现软件开发生命周期中的持续集成和发布。本文介绍版本的含义以及如何在函数计算控制台发布、查看和删除版本。

什么是版本

函数计算提供服务级别的版本控制功能，支持您为自己的服务发布一个或多个版本。版本相当于服务的快照，包括服务的配置、服务内的函数代码及函数配置，不包括触发器。当您发布版本时，函数计算会为服务生成快照，并自动分配一个版本号与其关联，以供后续使用。

如果没有版本功能，您在服务上的每次改动都会影响到生产环境，无法控制发布的时机。有了版本功能，您可以在测试稳定后发布服务版本，用稳定的版本来服务线上请求，并且可以继续在LATEST版本上开发测试。实现原理，如下图所示。



注意事项

- 新创建的服务，默认只有一个LATEST版本，在未发布任何版本前，LATEST版本是您拥有的唯一服务版本，LATEST版本不能被删除。
- 版本发布后，已发布的版本不可更改。且版本号单调递增，不会被重复使用。

前提条件

- 创建服务
- 创建函数

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务，然后在左侧导航栏，单击**版本管理**。
5. 在**版本管理**页面，单击**发布版本**，在发布服务的新版本面板，填写版本描述，然后单击**确定**。
   发布版本完成后，您可以在**版本管理**页面的版本列表查看刚才发布的版本。您还可以根据提示删除不需要的版本，将指定版本设置为别名的主版本或灰度版本，以及单击目标版本**操作**列的**日志库**跳转到日志服务SLS控制台查看日志情况。

> **说明** 删除一个版本只会删除该版本中的函数及配置，并不会删除指向此版本的别名或者触发器。因此，删除版本前请先移除指向此版本的别名和触发器，否则，如果调用指向当前版本的别名会提示错误。

更多信息

除了通过控制台，您还可以使用Serverless Devs为服务配置版本。更多操作，请参见Serverless Devs操作命令。

4.3. 管理别名

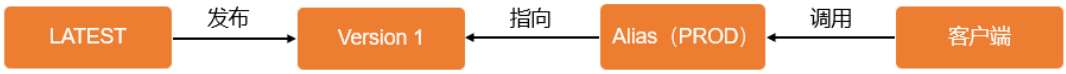
函数计算支持为服务版本创建别名。结合别名和版本功能，实现软件开发生命周期中的持续集成和发布。本文介绍别名的含义以及如何在函数计算控制台创建、更新、查看和删除别名。

什么是别名

函数计算支持为服务的版本创建别名。别名可以理解为指向特定版本的指针。您可以利用别名来轻松实现发布、回滚以及灰度发布等功能。别名无法脱离服务或版本单独存在。使用别名访问服务或函数时，函数计算会将别名解析为其指向的版本，调用方无需了解别名指向的具体版本。

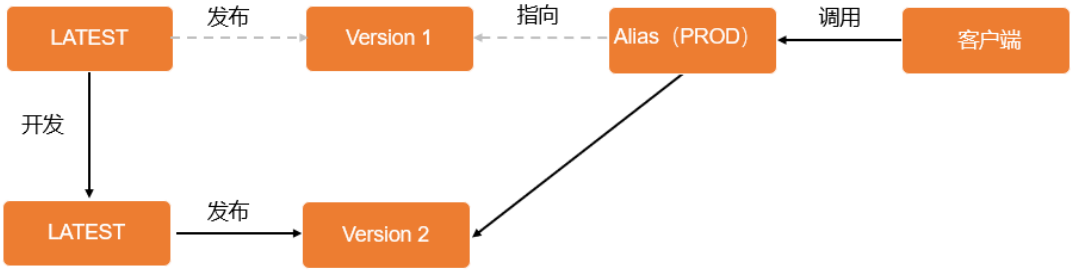
以HTTP触发器为例，如果没有别名，每次新版本上线，您需要手动修改HTTP触发器关联的版本号，在修改的过程中会影响客户端的使用。而如果使用别名进行版本管理，您可以实现版本的平滑升级。如下图所示，将别名PROD指向稳定的版本1。客户端直接通过别名PROD调用版本1下的函数。

发布版本1



版本1发布后，您可以继续在LATEST版本上开发新功能。由于客户端是通过别名调用对应版本下的函数，当需要发布新版本2时，只需要将别名PROD更新为指向版本2，此时，客户端调用时解析出的版本即为版本2，这样就可以完成版本的更新迭代。您也可以更改别名的指向，将别名PROD重新指向版本1回滚到之前的版本。通过这种方式发布，不会影响客户端的使用。

发布版本2



前提条件

- 创建服务
- 创建函数
- 发布版本

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务，然后在左侧导航栏，单击**别名管理**。
5. 在**别名管理**页面，单击**创建别名**。
6. 在创建别名面板，填写别名的相关信息，然后单击**确定**。

   参数说明如下。

参数	操作	本文示例
名称	填写自定义的别名名称。	alias
描述	填写该别名的描述信息。	无
主版本	选择比较稳定的主版本。	1

参数	操作	本文示例
启用灰度版本	若您需要将部分流量切到灰度版本，选择*是*。否则，选择*否*。	是
灰度版本	选择需要灰度发布的版本。当需要将部分流量切到灰度版本，该参数必选。	2
灰度版本权重	填写灰度版本的权重，即需要将多少比率的流量切换到灰度版本。当需要将部分流量切到灰度版本，该参数必选。	5

在**别名管理**页面，您可以看到刚才创建的别名。本示例中该别名指向版本1，灰度版本为版本2，灰度版本权重为5%。您还可以根据界面提示对已创建的别名进行编辑、删除不需要的别名或者单击目标别名**操作**列的**日志库**跳转到日志服务SLS控制台查看日志情况。

> **说明** 删除一个别名只会删除别名本身，并不会删除别名指向的版本，也不会删除指向此别名的触发器。

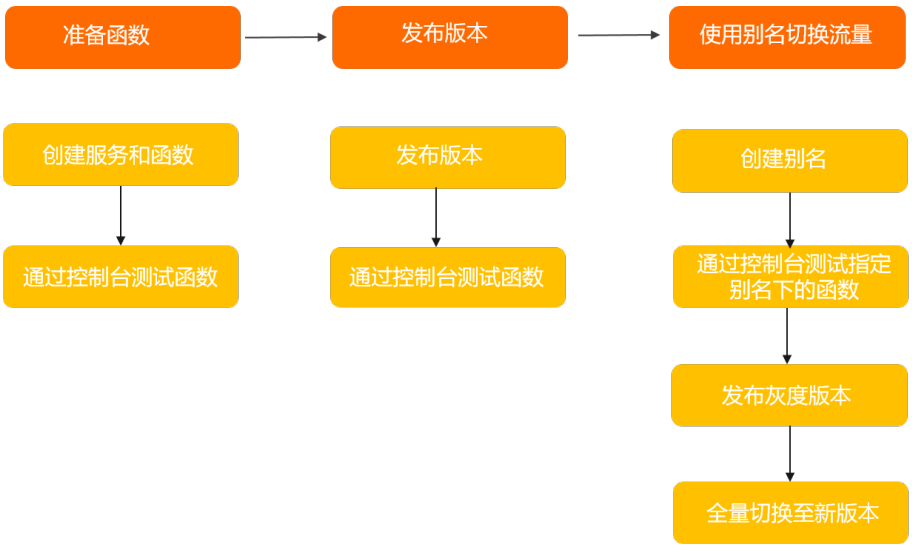
更多信息

除了通过控制台，您还可以使用Serverless Devs为服务配置别名。更多操作，请参见Serverless Devs操作命令。

4.4. 灰度发布示例

本示例介绍如何在函数计算控制台使用服务的版本和别名实现灰度发布功能。

灰度发布流程



前提条件

- 创建服务
- 创建函数

准备函数

当您初次创建一个服务以及该服务下的函数时，该服务的版本号为LATEST。您可以调试LATEST版本下的函数直至版本稳定运行。您可以通过控制台执行LATEST版本下的函数。

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 找到目标服务下的目标函数，单击函数名称。

5. 在函数详情页面，单击**函数代码**页签，在代码编辑器中修改代码为查看函数版本的代码，单击**保存并部署**，然后单击**测试函数**。

   查看函数版本的代码示例如下。

   Nodejs | Python | PHP | C#

   执行完成后，可以查看日志输出。从日志输出中，可以看到表示版本信息的字段qualifier的值为LATEST，即本次执行的函数为LATEST版本下的函数。



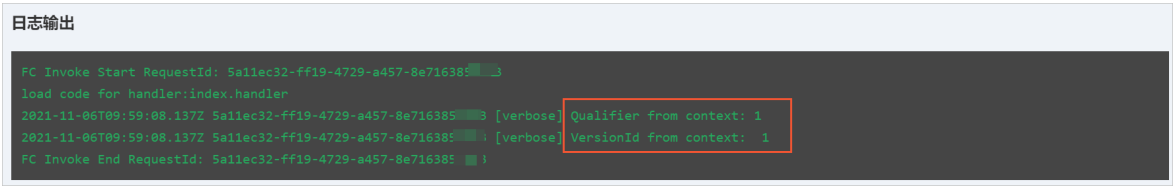
发布版本

当LATEST版本的服务稳定时，就可以发布该版本的服务，让稳定的版本来服务线上的请求。具体操作，请参见操作步骤。新版本发布后，您可以通过控制台执行新版本下的函数。

1. 在**服务列表**页面，单击目标服务。
2. 在**函数管理**页面，从**版本**下拉框选择新版本的版本号。



3. 单击目标函数名称，单击**函数代码**页签，然后单击**测试函数**。
   执行完成后，可以查看执行日志。从日志输出中，可以看到函数执行时的版本信息qualifier为1，解析出的versionId为1，即本次执行的函数为版本1下的函数。



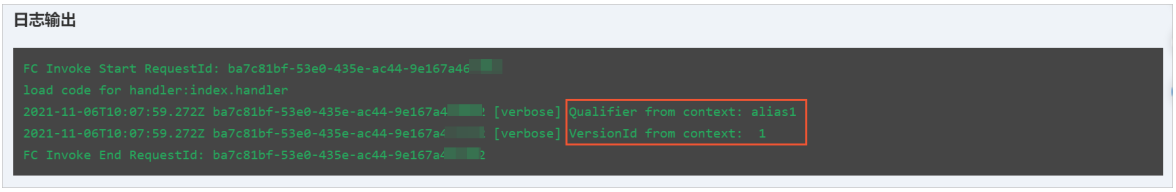
使用别名切换流量

新版本上线后，您可以创建一个别名，设置别名指向该版本。当该版本更新时，只需要将别名指向的版本更改为更新的版本。调用方无需关心服务的具体版本，只需要使用正确的别名即可。关于创建别名的具体步骤，请参见操作步骤。别名创建完成后，您可以通过控制台验证是否执行了正确版本的函数。

1. 在**服务列表**页面，单击目标服务。
2. 在左侧导航栏，单击**函数管理**，找到目标函数并单击目标函数。
3. 在函数详情页面，从**版本或别名**下拉列表中选择新创建的别名。



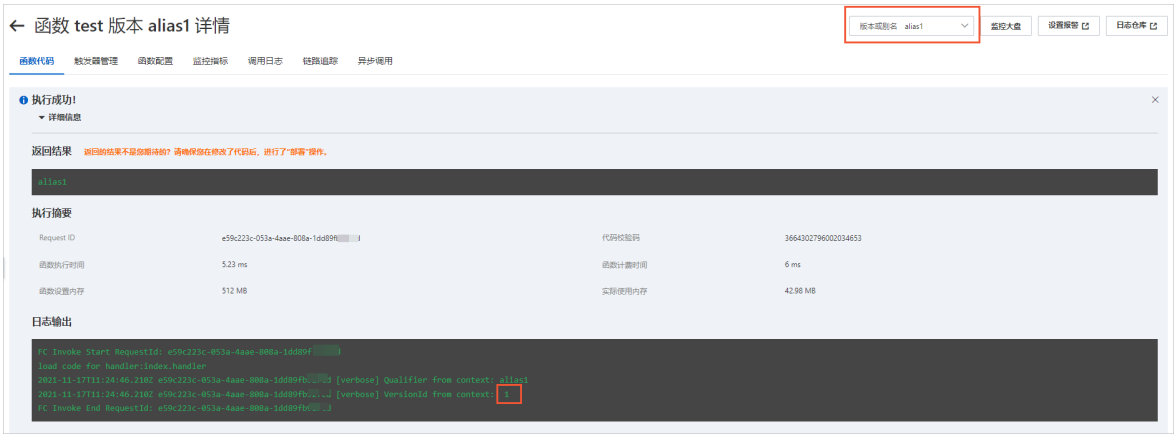
4. 单击**函数代码**页签，然后单击**测试函数**。
   执行完成后，可以查看日志输出。从日志输出中，可以看到函数执行时的版本信息qualifier为**alias1**，解析出的versionId为**1**，即本次执行的函数为别名**alias1**下的函数，该别名指向版本**1**。

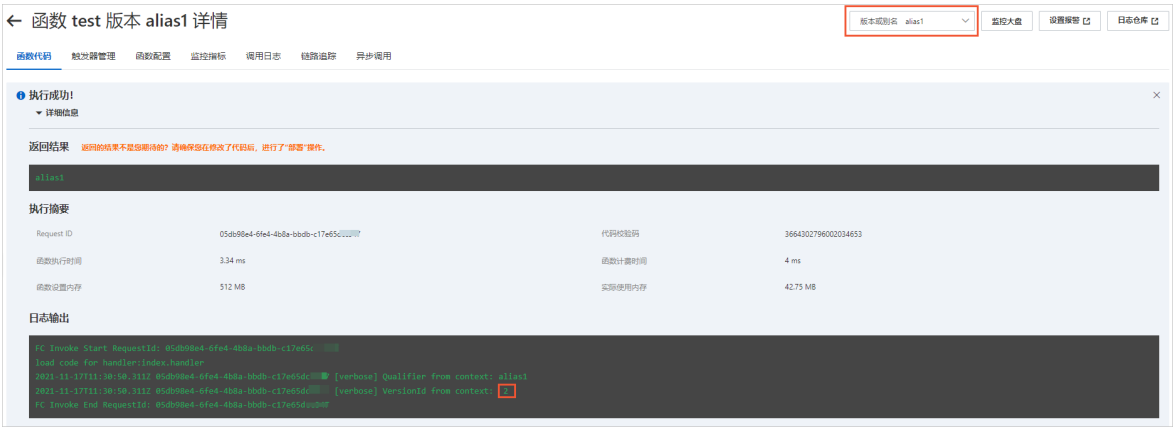


当新版本开发完成后，需要使用灰度版本帮助新版本的稳定发布。

1. 发布新版本。具体操作，请参见操作步骤。
   发布完成后，您可以在版本列表中查看新发布的版本。
2. 在左侧导航栏，单击**别名管理**。
3. 在别名列表中找到指向版本1的别名**alias1**，在其右侧**操作**列单击**编辑**。
4. 在编辑服务的别名面板，将新版本2设置为灰度版本，并填写灰度版本的权重。然后单击**确定**。

   设置完成后，您可以调用别名**alias1**验证是否已将部分线上流量切换到新版本2。本示例中，共调用别名**alias1**执行函数10次，结果中有3次调用版本了2，7次调用了版本1。





您也可以待灰度版本运行稳定后，将线上流量全部切换到新版本2。

常见问题

如何确定被调用的服务的版本？

使用灰度发布功能时，函数计算按照您指定的权重来分配流量，您可以通过以下方式来确定被调用的服务的版本：

- 通过context入参确定
  每次函数调用，context入参的service参数中会包括qualifier和versionId两个字段。
  - qualifier：调用函数时传入的版本信息，可以是版本号，也可以是别名。
  - versionId：函数执行时根据qualifier解析出的具体版本号。
- 通过同步函数调用响应确定
  同步函数调用的响应包含x-fc-invocation-service-version header，可以指示已调用的服务版本。

4.5. 配置网络

函数创建完成后，默认可以通过公网调用函数，无法通过VPC调用。如果您想让函数能够访问VPC内的资源，或允许指定的VPC来调用函数，则需手动为服务配置网络和权限。网络配置是服务级别的，即对该服务下的所有函数生效。本文介绍如何在函数计算控制台为服务配置网络。

网络访问能力

使用VPC功能会降低函数计算的冷启动效率，建议非必要请勿配置该功能。您可优先选择使用RAM授权方式访问资源，例如表格存储Tablestore。

通过网络地址访问函数或通过函数访问网络地址时，会产生流量。流量可分为以下类型：

- 公网流量：访问Internet地址，例如阿里云官网、淘宝网和云服务的公网地址。
- 私网流量：访问云产品内网地址，例如OSS内网Endpoint、ECS默认DNS的地址。
- VPC流量：访问用户VPC内的地址，例如RDS地址、NAS地址和VPC中ECS的内网地址。

根据对网络的不同设置，函数有以下网络访问能力，您可按需设置：

- 函数出流量：是否允许函数访问公网或VPC内资源的出流量，对应的配置包含**允许访问 VPC**和**允许函数访问公网**。函数出流量

网络配置	说明
仅允许函数访问公网	通过函数网络访问公网和内网，禁止访问VPC流量。所需的网络配置如下： ○ 设置**允许访问 VPC**为**否**。 ○ 设置**允许函数访问公网**为**是**。
仅允许函数访问VPC	通过用户VPC访问公网、内网和VPC流量，适用于PrivateZone、NAT网关和函数绑定VPC等场景。所需的网络配置如下： ○ 设置**允许访问 VPC**为**是**，并配置允许函数访问的VPC信息。 ○ 设置**允许函数访问公网**为**否**。
允许函数既能访问公网，也能访问VPC	通过函数网络访问公网和内网，通过用户VPC访问VPC流量。所需的网络配置如下： ○ 设置**允许访问 VPC**为**是**，并配置允许函数访问的VPC信息。 ○ 设置**允许函数访问公网**为**是**。
既不允许函数访问公网，也不允许访问VPC	通过函数网络访问内网，禁止访问公网和VPC流量。所需的网络配置如下： ○ 设置**允许访问 VPC**为**否**。 ○ 设置**允许函数访问公网**为**否**。

- 函数入流量：是否允许通过公网地址或VPC地址访问函数的入流量，对应的配置为**仅允许指定 VPC 调用函数**。函数入流量

网络配置	说明
允许通过公网和VPC同时访问函数	函数创建完成后，默认可以通过公网和VPC调用函数。默认的网络配置如下： ○ 设置**仅允许指定 VPC 调用函数**为**否**。
仅允许通过VPC访问函数	允许通过指定的VPC调用函数，并禁止通过公网调用函数。所需的网络配置如下： ○ 设置**仅允许指定 VPC 调用函数**为**是**，并配置允许调用函数的VPC信息。

可用区注意事项

如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算支持的可用区的交换机，并在函数计算的服务的VPC配置中设置此交换机ID。由于同一VPC内不同交换机之间私网互通，因此函数计算可以通过该交换机访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。

函数计算支持的可用区 ❯

前提条件

- 创建服务
- （可选）创建网络资源
  如果您未创建相关资源，可在具体配置时，选择**自动配置**。否则，需按以下文档操作指导提前创建相关资源：
  - 创建专有网络和交换机
  - 创建安全组

配置网络和角色

函数计算访问VPC的配置和权限的配置是服务级别的，为一个服务配置了访问VPC的能力后，此服务下的所有函数都可以访问VPC。

1. 登录函数计算控制台。
2. 在顶部菜单栏，选择地域。
3. 在左侧导航栏，单击**服务及函数**。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在编辑服务页面的**角色配置**区域，选择**服务角色**，授予服务访问VPC的权限。

   推荐您按照最小权限原则为您提供的`AliyunFCDefaultRole`角色。如需更细粒度的权限管控，请参见权限策略及示例。

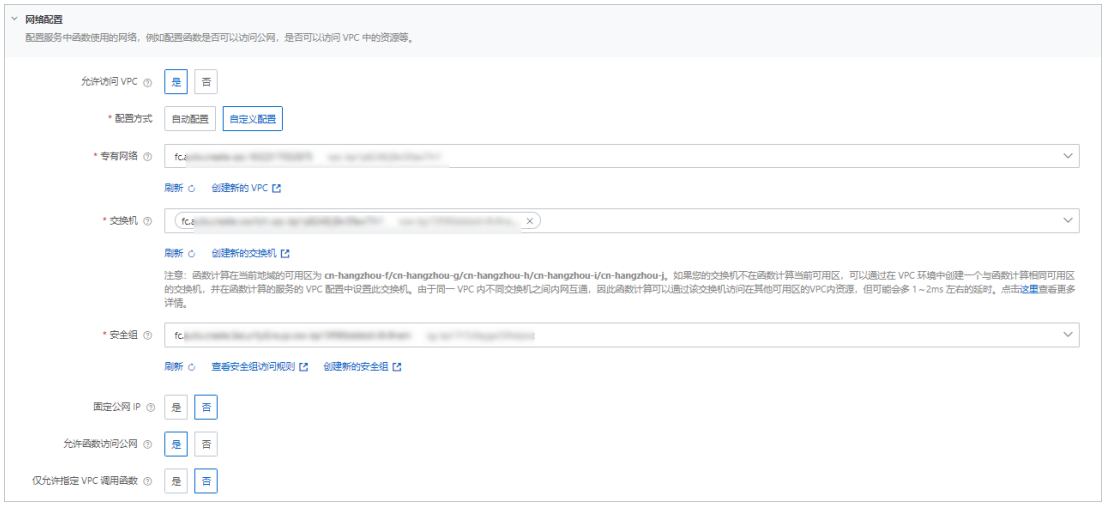
6. 在**网络配置**区域，根据实际情况修改以下网络配置参数。
   - **允许访问 VPC**：是否允许函数访问VPC内资源。取值说明如下：

- **是**：允许函数访问VPC内资源。选择**是**后，则需配置**配置方式**，取值说明如下：
  - （推荐）**自动配置**：函数计算会为您自动创建专有网络、交换机和安全组等资源，无需您手动创建。创建完成后，也可再次按需修改。



> **说明** 自动创建的网络资源均以fc.auto.create开头。

  - **自定义配置**：您需手动选择已有的网络资源，请确保您已提前创建好相关资源。



    涉及的参数如下：
    - **专有网络**：在列表中选择要访问的VPC ID。
    - **交换机**：在列表中选择交换机ID，至少需选择一个交换机ID。
      该字段限定了函数计算可以访问的子网，建议设置两个或多个交换机ID。当一个可用区出现故障或IP地址不足时，您的函数可以在其他子网下运行。
    - **安全组**：在列表中选择安全组ID。
      此安全组是函数计算所在的安全组，安全组限定了函数计算在VPC中的出入站规则。需要设置您的VPC所在安全组的入站规则为允许函数计算所在的安全组访问。否则，函数计算无法顺畅地访问您的VPC内资源。
  - **否**：不允许函数访问VPC内资源。

- **固定公网 IP**：是否通过NAT网关和弹性公网IP地址获得一个固定的公网出口IP地址。具体信息，请参见配置固定公网IP地址。
- **允许函数访问公网**：是否允许函数访问公网。取值说明如下：
  - **是**：允许函数访问公网。
  - **否**：不允许函数访问公网。
- **仅允许指定 VPC 调用函数**：是否允许指定的VPC调用函数。取值说明如下：
  - **是**：允许指定的VPC调用函数。需注意以下事项：
    - 同一个服务最多绑定20个VPC。
    - 设置仅允许指定VPC调用函数后，使用触发器调用函数不受影响。
    - VPC绑定后对服务的所有版本和别名生效。
    - 设置允许指定VPC调用函数后，会拒绝来自公网和其他VPC的调用请求，`StatusCode`为403，`ErrorCode`为`AccessDenied`，错误信息为`Resource access is bound by VPC: VPCID`。
    - 只允许使用私网HTTP接入点绑定VPC，不允许使用公网接入点和私网HTTPS接入点绑定VPC。

- 否：不允许指定VPC调用函数，仅允许通过公网调用函数。

7. 单击**保存**。
   页面跳转至**服务详情**页面，您可查看到刚才保存的服务配置信息。

常见问题

- 为什么函数计算无法成功接入VPC调试？
  如果您的服务中已经设置允许函数访问VPC，却无法成功接入VPC，可能是以下原因：
  - 交换机所在的子网故障或子网中的IP地址用尽，您可以在配置VPC时提供两个或者多个交换机ID，如果一个可用区出现故障，您的函数可以在其他可用区运行，提高容错能力。
  - 安全组配置错误，请您按照以下要求配置安全组：
    - 需要设置VPC中的安全组的入站规则为允许函数计算所在的安全组访问。
    - 安全组出口方向需要允许ICMP协议，函数计算会通过ICMP协议检查VPC网络联通性。

    安全组的具体配置步骤，请参见添加安全组规则。
- 如果我新增网络资源时，发现资源不足该怎么办？
  如果您在创建VPC网络资源时，自动配置提供的网络前缀长度为24，可用IP地址数量是252个，如果实例数量过多，有可能会超出限制。此时，您需手动调整交换机IP地址网段，以及对应的安全组。

故障排查

函数计算无法在设置vpcConfig时对访问VPC的权限进行检测，而需要在执行函数时检测，因此在通过InvokeFunction接口调用函数时会引入一些新的错误类型。下表描述了接入VPC时一些常见的错误，以便您迅速排查问题。

错误码	状态码	问题原因	解决方案
InvalidArgument	400	函数计算不支持您提供的vSwitchId所在的可用区。	重新设置vSwitchId。具体信息，请参见可用区注意事项。
		未找到vpcConfig中的vpcId、vSwitchIds或securityGroupId对应的资源。	检查vpcConfig参数设置。
		设置的交换机或安全组不在相应的VPC中。	检查vpcConfig参数设置，确保vSwitchId和securityGroupId对应的资源在vpcId对应的VPC中。
AccessDenied	403	未提供对ENI的操作授权。	检查服务的权限。更多信息，请参见授予函数计算访问其他云服务的权限。
ResourceExhausted	429	交换机的网段内可用IP不足，函数计算无法创建更多的ENI。	创建一个网段更大的交换机，并更新vpcConfig的vSwitchId参数。 **说明** 建议您使用 `/24` 或 `/16` 网段。

后续步骤

如需访问VPC内的数据库，具体步骤，请参见访问数据库。

4.6. 配置固定公网IP地址

函数计算访问数据库、微信小程序或其他第三方服务时，需通过配置白名单来访问，您可以使用函数计算的固定公网IP地址来完成白名单的验证。本文以首次配置和增加固定公网IP地址两种场景为例介绍如何在函数计算控制台配置函数出口方向的固定公网IP地址。

注意事项

- 固定IP地址功能需配合专有网络VPC的公网NAT网关来完成。通过创建一个公网NAT网关，并为其绑定一个弹性公网IP地址和添加一个SNAT条目，允许VPC内的实例通过此弹性公网IP地址访问公网。此时，在公网接收端查询到的客户端IP地址为该NAT网关绑定的弹性公网IP地址。
- NAT网关和弹性公网IP地址均只能在某个地域的固定可用区购买，如果NAT网关、弹性公网IP与函数计算的服务所在的可用区不一致，您必须通过设置交换机来实现跨可用区公网互通。更多信息，请参见可用区注意事项。
- 弹性公网IP地址在同一地域下的所有服务间共享。
- RAM用户使用固定IP地址功能前，需先为其授予 `AliyunECSFullAccess` ， `AliyunVPCFullAccess` ， `AliyunEIPFullAccess` 和 `AliyunNATGatewayFullAccess` 权限。

计费说明

配置固定公网IP地址可能会创建NAT网关和弹性公网IP地址导致费用问题。具体信息，请参见公网NAT网关计费和弹性公网IP地址计费。

首次配置固定公网IP地址

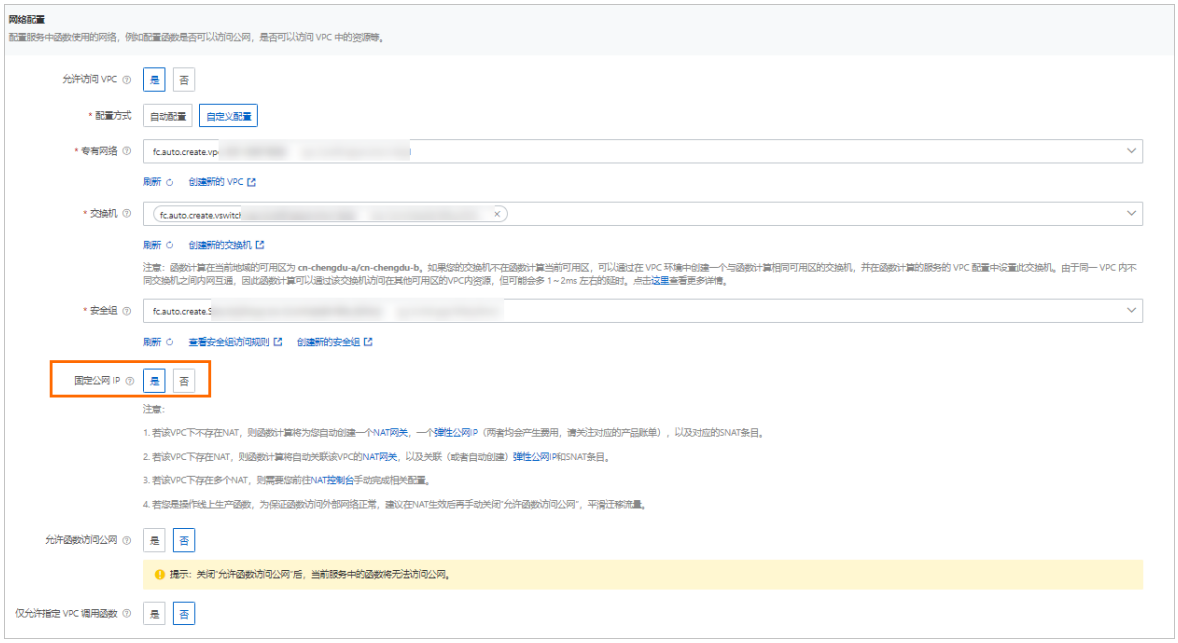
前提条件

- 您已完成函数访问VPC资源相关权限和网络的配置。具体操作，请参见配置网络和角色。
- 您已配置**允许访问 VPC**为**是**。具体操作，请参见配置网络。
- 您已配置**允许函数访问公网**为**是**。具体操作，请参见配置网络。

> **说明** 为保证函数访问外部网络正常，建议您先配置**允许函数访问公网**为**是**，固定IP生效之后再配置**允许函数访问公网**为否，平滑迁移流量。

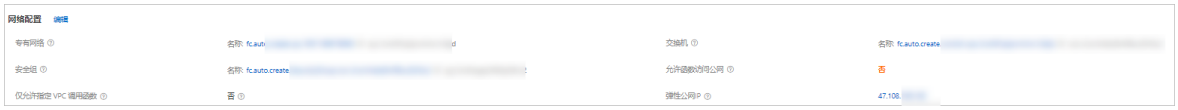
操作步骤

1. 登录函数计算控制台。
2. 在顶部菜单栏，选择地域。
3. 在左侧导航栏，单击**服务及函数**。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在**网络配置**区域，参数**固定公网 IP**选择**是**。



> **说明** 执行此操作后，函数计算可能会自动创建NAT网关相关资源，例如NAT网关、弹性公网IP和SNAT条目等，其名称均以 `fc_auto_create` 开头。

6. 单击**保存**，在弹出的**固定IP配置**对话框，阅读提示信息，单击**确定**，然后再单击**保存**。
   3~5分钟后前往**服务详情**页面，您可查看到刚才保存的服务配置信息；或前往专有网络控制台，您可以看到刚才关联或自动创建的公网NAT网关相关资源信息。



结果验证

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在**函数管理**页面，单击目标函数名称。
6. 单击**函数代码**页签，在代码编辑器中编写代码。

   Python 2.7代码示例如下：

```
# -*- coding: utf-8 -*-
import logging
import requests
def handler(event, context):
  logger = logging.getLogger()
  # Ask myip.ipip.net, what is my IP?
  r = requests.get('https://myip.ipip.net')
  clientIP = r.content.split()[1]
  logger.info('Client IP ' + clientIP)
  return clientIP
```

7. 完成函数编写并部署后，单击**测试函数**。
   执行成功后，查看返回的客户端IP地址，即配置的固定IP地址。



增加固定公网IP地址

如您需要增加其他的固定IP地址，步骤操作如下：

1. 在**网络配置**区域，单击**固定IP**参数下方的**创建新的弹性公网IP**。



   页面自动跳转到专有网络控制台。

2. 在专有网络控制台，手动创建一个弹性公网IP地址并绑定到对应的NAT网关。具体步骤，请参见申请EIP。

   绑定成功后，前往**服务详情**页面，您可查看到刚才保存的服务配置信息。



更多操作

如您需要关闭固定IP地址功能或删除已配置的固定IP地址，请先配置**允许函数访问公网**为**是**，然后前往专有网络控制台手动删除VPC对应的所有NAT实例。

4.7. 配置NAS文件系统

阿里云文件存储NAS（Apsara File Storage NAS）是一种分布式的网络文件存储系统，为ECS、HPC、Docker、BatchCompute等提供安全、高性能、高可靠、简单易用的文件存储服务。

背景信息

阿里云函数计算支持与文件存储NAS无缝集成。您可以在函数计算的服务上配置NAS，包括NAS的地域、挂载点、分组等信息。配置成功后，该服务下的函数就可以像访问本地文件系统一样访问指定的NAS文件系统。

使用NAS作为函数计算的挂载点优势如下所示：

- 可以将临时文件存储到NAS中，临时文件大小不受实例本地磁盘空间限制。
- 多个函数可以共用一个NAS，实现文件共享。

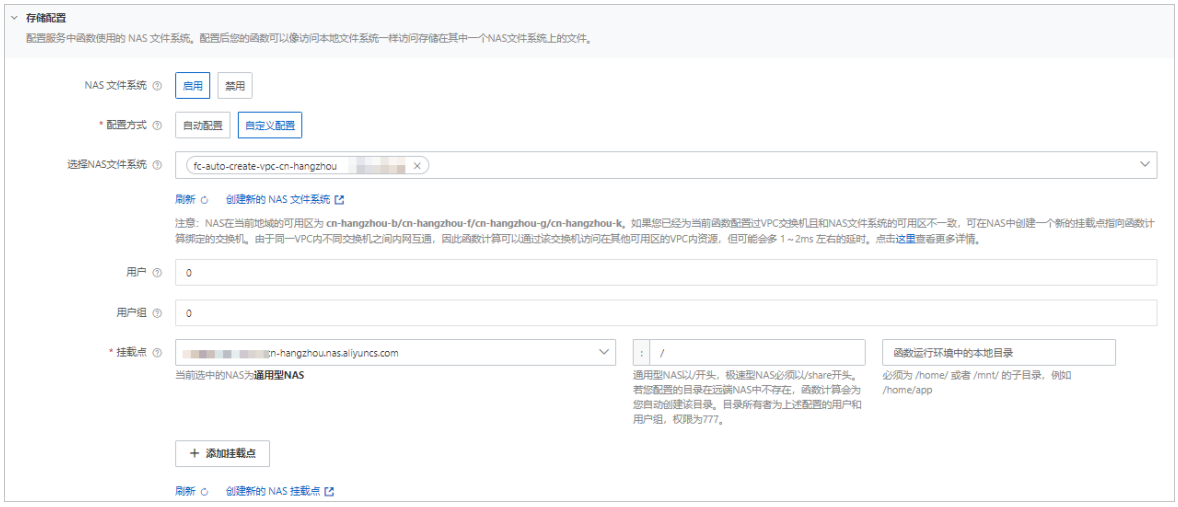
前提条件

- 函数计算
  - 配置网络
    NAS目前只能在私有的VPC环境才能添加挂载点，因此您必须确保配置正确的VPC才能访问指定的NAS文件系统。
- 文件存储NAS
  - 通过控制台创建通用型NAS文件系统
  - 添加挂载点

配置NAS

函数计算的NAS配置是服务级别的，为一个服务配置了NAS挂载点后，此服务下的所有函数都可以访问指定NAS文件系统上的文件。

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在编辑服务页面的**存储配置**区域，配置以下参数，然后单击**保存**。



- **NAS 文件系统**：是否启用NAS文件系统。取值说明如下：
  - **启用**：启用NAS文件系统。
  - **禁用**：不启用NAS文件系统。
- **配置方式**：NAS文件系统的配置方式。取值说明如下：
  - **自动配置**：系统自动为您配置NAS文件系统。

    > **说明**
    > - 选择**自动配置**前，需先允许函数访问VPC内资源，并选择一个**专有网络**、**交换机**和**安全组**。如未选择，系统将会为您自动创建以上资源并绑定到当前服务。关于费用详情，请参见产品计费。
    > - 选择**自动配置**，系统将为您创建一个新的通用型NAS文件系统。如您再次选择**自动配置**，系统查询到前一次选择**自动配置**时创建的通用型NAS文件系统后将不会再重新创建。关于费用详情，请参见通用型NAS计费说明。

  - **自定义配置**：您需要手动配置如下参数才能配置NAS文件系统。
    - **选择NAS文件系统**：选择已创建的NAS文件系统。您也可以单击下方的**创建新的 NAS 文件系统**跳转到文件存储控制台手动创建新的NAS文件系统。
    - （可选）**用户**：在文本框中填写自定义的用户ID。如您不输入任何值，系统将使用root用户ID，即 `UID=0` 。更多信息，请参见NAS用户和用户组。
    - （可选）**用户组**：在文本框中填写自定义的用户组ID。如您不输入任何值，系统将使用root用户组ID，即 `GID=0` 。更多信息，请参见NAS用户和用户组。
    - **挂载点**：在列表中选择与当前服务配置的专有网络和交换机匹配的NAS挂载点，同时填写**远端 NAS 中的目录**和**函数运行环境中的本地目录**。更多信息，请参见地址配置。
      - **远端 NAS 中的目录**：通用型NAS的该目录必须以*/*开头，极速型NAS的该目录必须以*/share*开头。若您配置的目录在远端NAS中不存在，函数计算将会为您自动创建该目录。目录所有者为文件系统的用户和用户组，权限等级为 `777` 。
      - **函数运行环境中的本地目录**：必须为*/home*或者*/mnt*的子目录。

> **说明**
>
> - 配置**挂载点**之前，需先允许函数访问VPC内资源，并选择同一地域下的一个**专有网络**和**交换机**。具体信息，请参见配置网络。
> - 一个服务最多可以挂载5个NAS挂载点。

NAS用户和用户组

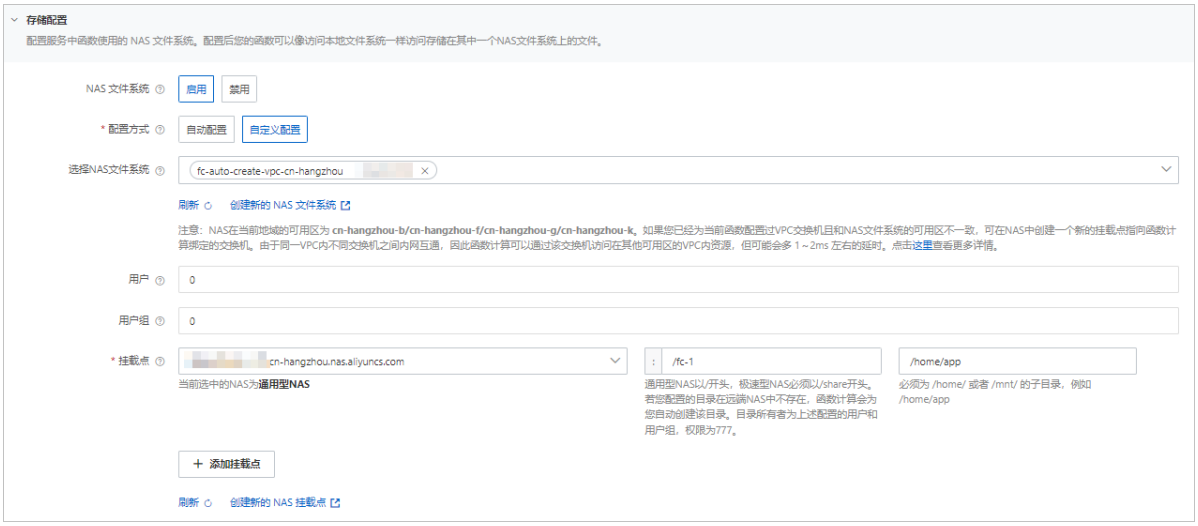
在配置函数的NAS挂载时，首先需要配置UserID（用户ID）和GroupID（用户组ID），这两个值等同于文件系统中的用户和组的概念，请根据需求设置文件的拥有者和相应的组权限，确保文件读写权限一致。

UserID和GroupID取值范围从0到65534。UserID和GroupID值配置可选，如果不填写UserID，系统会使用root用户ID，即 `0` 作为UserID值；如果不填写GroupID，系统会使用root用户组ID，即 `0` 作为GroupID值。

函数计算使用非root权限执行用户代码时，如果需要挂载远端NAS的子目录，需确保函数实例中执行用户具有子目录的读写权限。建议您选择以下两种方法中任意一种进行操作：

- 使用NAS实例挂载远端NAS的根目录，具体操作，请参见配置NAS。挂载成功后使用以下示例创建子目录，如*fc-1*。然后将NAS挂载点中的**远端 NAS 中的目录**由根目录修改为创建的子目录（由/修改为*/fc-1*），实现远端挂载子目录。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
def handler(event, context):
  print('uid : ' + str(os.geteuid()))
  print('gid : ' + str(os.getgid()))
  # 请手动更改为 NAS 挂载的本地目录
  local_nas_dir = "/home/app"
  # 请手动更改为目标子目录名
  target_sub_dir = "fc-1"
  # 在 FC 中利用当前实例执行用户权限创建目标目录
  new_dir = local_nas_dir + '/' + target_sub_dir + '/'
  print('new_dir : ' + str(new_dir))
  os.mkdir(new_dir)
  return 'success'
```



- 使用云服务器ECS挂载NAS，具体操作，请参见使用ECS挂载NAS文件系统。挂载成功后创建子目录，然后执行 `chmod 777` 为子目录授权。

> 说明
> - 系统默认的用户和用户组无文件的读写权限，因此强烈建议您将UserId和GroupId设置为具体的值，即1～65534的任意数字。此时，该服务下不同函数可以共享这些文件资源。
> - 上传至NAS的文件权限与本地文件权限完全相同。

地址配置

每个**挂载点**配置由**远端 NAS 中的目录**和**函数运行环境中的本地目录**组成。远端目录与本地目录结合，形成了从NAS文件系统中的某个目录到本地文件系统中的一个目录的映射。

- **远端 NAS 中的目录**
  远端NAS中的目录描述了服务需要访问的NAS文件系统的目录，由挂载点和绝对目录两部分组成。挂载点可以通过NAS控制台来添加。将挂载点和绝对目录拼接就可以得到远端目录。例如，如果NAS文件系统的挂载点是*xxxx-nas.aliyuncs.com*，您希望被访问的绝对目录是*/workspace/document*，对应完整的远端目录就是*xxxx-nas.aliyuncs.com:/workspace/document*。
  您可以登录NAS控制台，在文件系统列表中，单击目标文件系统，然后单击**挂载使用**，在挂载点列表中获取挂载点。
- **函数运行环境中的本地目录**
  函数运行环境中的本地目录是指本地文件系统的挂载点，请不要使用通用的Linux和Unix系统目录，例如*bin*、*opt*、*var*、*dev*等挂载NAS。函数计算允许您使用*mnt*、*home*等非系统目录挂载NAS。

更多信息

除了通过控制台，您还可以使用Serverless Devs为服务挂载NAS系统。更多操作，请参见使用s nas相关命令挂载硬盘。

4.8. 配置日志

您可以将函数调用执行的日志存储至阿里云日志服务SLS，再根据日志服务中存储的函数日志来执行代码调试、故障分析、数据分析等操作。本文介绍如何在函数计算控制台配置日志功能，以及查看函数调用日志的方法介绍。

背景信息

日志服务是阿里云提供的针对日志类数据的一站式服务。通过日志服务存储函数日志，需要在函数对应的服务中配置日志项目和日志库，并授予该服务访问日志服务的权限。函数日志会打印到配置的日志库中，同一地域的同一个服务下的所有函数日志都会打印到同一个日志库中。

计费说明

配置日志功能时，函数计算可能会自动为您创建**日志项目**和**日志库**，并以按量付费模式计费。计费详情，请参见计费项。

前提条件

- 函数计算
  - 创建服务
- （可选）日志服务SLS
  - 创建日志项目和日志库

配置日志功能

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，找到目标服务。在其**操作**列，单击**配置**。
5. 在编辑服务的**日志配置**区域，配置以下参数，然后单击**保存**。



说明

- 如果您在创建服务时已启用日志功能，此处**日志项目**和**日志库**将自动填充函数计算自动创建的以 `aliyun-fc-cn-<region_id>` 开头的**日志项目**和该**日志项目**下默认的**日志库**。如系统查询到该地域下前一次自动创建的**日志项目**，将直接使用，不再重新创建。
- 如果您在创建服务时未启动日志功能，此处需手动选择**日志项目**和**日志库**。

参数	是否必填	说明
日志功能	是	是否启用日志服务，取值说明如下： - **启用**：启用后，您可以查看函数的调用日志，函数的调用日志将通过日志服务存储和查询。 **说明** 启用日志功能时，需要在当前页面的**角色配置**区域选择一个可以访问日志服务的角色。服务角色的更多信息，请参见授予函数计算访问其他云服务的权限。 - **禁用**：不启用日志功能。
日志项目	是	指定存储函数调用日志的目标日志项目。
日志库	是	指定存储函数调用日志的目标日志库。
日志分割规则	否	是否启用日志分割规则。取值说明如下： - **启用**：启用后，函数计算将按日志分割规则进行切分，切分后的日志块将被写入到日志服务。该规则将按照行首是否带有日期进行切分，例如您的日志行首是2021-10-10，则该日志将被认为是一块日志的首行，首行和接下来连续不带日期的日志将被作为整体写入到日志服务。 - **禁用**：不启用日志分割规则。

参数	是否必填	说明
请求级别指标	否	是否启用函数计算提供的调用分析功能来查看请求级别的指标。取值说明如下： ○ **启用**：启用后，函数每次调用执行的指标信息将被收集，并被投递到您在日志配置时选择的Logstore。您可以结合业务日志与分析日志来创建自定义监控大盘，即Grafana大盘。更多信息，请参见基于请求级别指标创建Grafana大盘。 ○ **禁用**：不启用函数计算提供的调用分析功能来查看请求级别的指标。
实例级别指标	否	是否启用实例级别指标功能。取值说明如下： ○ **启用**：启用后，您可查看函数实例维度的性能指标，对函数实例进行实时监控和性能数据采集，并进行可视化展示，为您提供函数实例端到端的监控排查路径。更多信息，请参见实例级别指标。 ○ **禁用**：不启用实例级别指标。

> **说明** 您也可以在函数详情页面的**调用日志**页签，单击**一键启用**来启用日志服务功能。函数计算将自动为您创建以 `aliyun-fc-cn-<region_id>` 开头的**日志项目**和该**日志项目**下默认的**日志库**。如系统查询到该地域下前一次自动创建的**日志项目**，将直接使用，不再重新创建。

查看调用日志

在函数详情页面，单击**调用日志**，查询当前函数的调用记录。

> **说明** 您可以在页面上方切换版本或别名，查询指定函数版本或别名的日志。

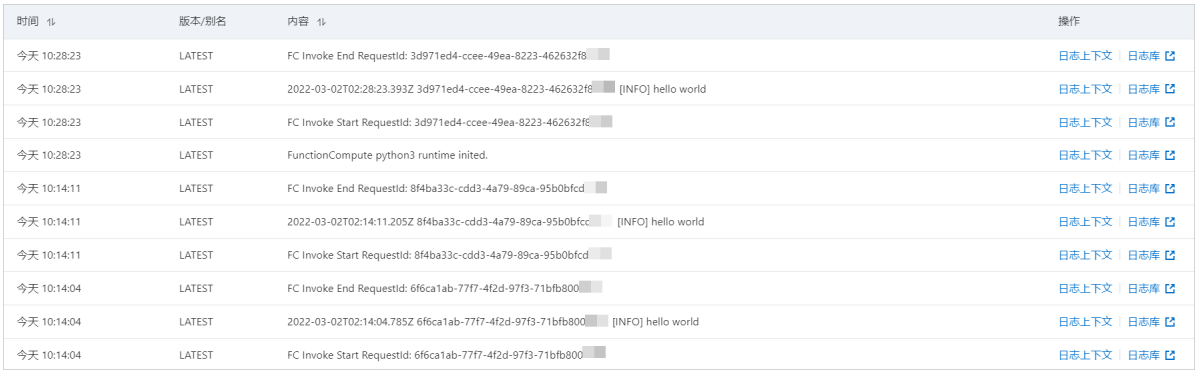
调用请求列表页签或**关键词搜索**页签均可以查看函数调用日志的内容。二者的区别如下所示：

- **调用请求列表**
  列表显示函数调用时间、请求ID、调用结果、实例ID和函数的版本、别名等信息。如下图所示：

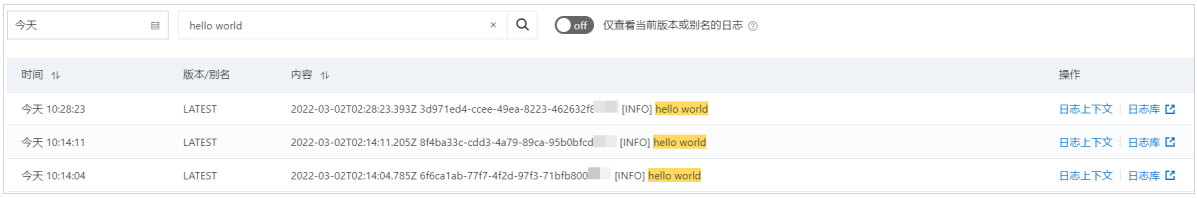


  - 您可以单击对应的**Request ID**，查看请求详情和日志详情。
  - 您可以单击对应的**实例 ID**，查看实例的指标使用情况，例如CPU使用情况、内存使用情况和网络流量等。
  - 您可以单击**操作**列的**查看日志**查看日志详情，或单击**操作**列的**高级日志**跳转到日志服务控制台查询和分析日志。
- **关键词搜索**

列表显示函数调用时间、日志内容、和函数的版本、别名等信息，如下图所示。您可以单击**操作**列的**日志上下文**查找这条日志的若干条上下文日志，或单击**操作**的**日志库**跳转到日志服务控制台查询和分析日志。



您可以在搜索框输入关键词查询您所需的日志信息。例如您需要搜索包含文本 `hello world` 的日志，可以在搜索框输入关键词 `hello world`，搜索结果为所有包含该关键词的日志行，如下图所示。更多关于查询关键词的语法，请参见查询语法。



更多信息

除了通过控制台，您还可以使用Serverless Devs来配置相应日志服务资源并查看函数调用的日志：

- Serverless Devs操作命令
- 命令行模式使用指南

4.9. 管理标签

函数计算支持将相同作用的服务资源通过标签（Tag）归类，便于搜索和资源聚合。您还可以通过标签功能将服务进行分组，可以实现不同角色对不同分组的服务拥有不同的操作权限。本文介绍标签功能的使用说明、使用限制以及如何在函数计算控制台操作标签等。

使用说明

- 标签与服务资源是多对多的关系。
- 每个标签都由一对键值对（Key:Value）组成。
- 标签可以是授权的一个条件，为一个确定范围内的资源做精细粒度授权。
- 服务下的资源，版本、别名、函数和触发器均继承服务的标签。
  - 只要API的调用上填写了服务，都支持标签鉴权。
  - 不同版本的服务都是相同的标签meta，即标签修改影响的标签鉴权针对的是所有版本和别名的服务。

使用限制

- 键（Key）的最大长度：64个Unicode字符，区分大小写。
- 值（Value）的最大长度：128个Unicode字符，区分大小写。
- 同一个资源上的同一个键只能有一个标签。如果您尝试添加的标签的键与现有的标签的键相同，现有标签值会更新为新值。
- 每个资源的最大标签数：20。
- 键（Key）不支持 `aliyun` 或 `acs:` 开头，不允许包含 `http://` 和 `https://`，不允许为空字符串。
- 值（Value）不允许包含 `http://` 和 `https://`，不允许为空字符串。
- 每个地域中的标签信息不互通。例如在华东1（杭州）地域创建的标签在华东2（上海）地域不可见。

新建标签

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在左侧导航栏，单击**服务详情**。
6. 在**基础信息**区域，将鼠标移至**标签**后的图标上，然后在弹出的对话框单击**+添加**。



7. 在编辑服务标签面板，填写**标签键**和**标签值**，然后单击**确定**。

   重复操作该步骤，可填写多个键值对，为目标服务添加多个标签。



完成新建标签后，将鼠标再次移至**服务详情**页面的图标上，您可以看到刚才新建的标签。



更新标签

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在左侧导航栏，单击**服务详情**。
6. 在**基础信息**区域，将鼠标移至**标签**后的图标上，然后在弹出的对话框单击**编辑**。



7. 在编辑服务标签面板，填写新的**标签键**和**标签值**或修改已有**标签键**的**标签值**，然后单击**确定**。



删除标签

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在左侧导航栏，单击**服务详情**。
6. 在**基础信息**区域，将鼠标移至**标签**后的 图标上，然后在弹出的对话框单击**编辑**。



7. 在编辑服务标签面板，找到目标**标签键**和**标签值**所在行，单击其右侧的删除图标 ，然后单击**确定**。

使用标签对服务分组授权

假设您创建了10个函数计算的服务，需要将5个服务授权给dev团队，另外5个服务授权给ops团队。您希望每个团队只能查看被授权的服务，未被授权的服务则不允许查看。此时，您可以通过标签功能将各团队进行分组，然后给不同分组的团队授予不同的权限。您需给其中5个服务添加一对标签，标签键是*team*，标签值是*dev*；另外5个服务添加另一对标签，标签键是*team*，标签值是*ops*。

1. 将5个授权给dev团队的服务打上*team:dev*标签，5个授权给ops团队的服务打上*team:ops*标签。详细操作，请参见新建标签。
2. 创建RAM用户。

3. 创建用户组。

   创建dev和ops两个用户组。

4. 为用户组添加RAM用户。

   将不同的RAM用户添加到相应的用户组下。

5. 为两个用户组授予不同的权限。

   权限策略分为系统权限策略和自定义权限策略，根据实际场景选择合适的权限策略：

   - 为用户组授予系统权限策略。
     详细操作，请参见为用户组授权。
   - 为用户组授予自定义权限策略。

     a. 创建自定义权限策略。
     假设给dev团队创建的自定义策略名称为*policyForDevTeam*，策略示例如下。

```
{
    "Statement": [
        {
            "Action": "fc:*",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "fc:tag/team": "dev"
                }
            }
        },
        {
            "Action": "fc:ListServices",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "fc:GetResourceTags",
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

     假设给ops团队创建的自定义策略名称为*policyForOpsTeam*，策略示例如下。

```
{
    "Statement": [
        {
            "Action": "fc:*",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "fc:tag/team": "ops"
                }
            }
        },
        {
            "Action": "fc:ListServices",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "fc:GetResourceTags",
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

b. 为用户组授权。
选择权限时请选择刚创建好的自定义策略。

完成授权后，处于dev用户组的RAM账号只能操作标签为*team:dev*的服务，处于ops用户组的RAM账号只能操作标签为*team:ops*的服务。

4.10. 服务管理FAQ

4.10.1. 什么是VPC?

专有网络VPC（Virtual Private Cloud）是为私人使用而构建的隔离云网络。它允许您在虚拟网络环境中逻辑地隔离云资源。所有函数都运行在函数计算拥有的VPC网络环境中。由于VPC网络隔离的性质，函数计算在缺省VPC配置的情况下不能访问您的私有VPC资源。

4.10.2. 如何用函数访问私有VPC资源?

函数计算允许您配置您的函数去访问特定的VPC。您可以授予函数计算操作弹性网络接口（ENIs）的权限，并提供特定的VPC配置信息，包括VPC ID、交换机ID和安全组ID。一旦VPC配置被启用，您的函数就运行在特定的VPC内部。

4.10.3. 函数可以访问多少个VPC?

每个函数最多可以配置一个VPC。如果您的函数需要访问一个以上的VPC内的资源，您可以将您的VPC拼接成一个大VPC，并授予您的函数访问这个大VPC的权限。

4.10.4. 我在VPC环境里如何访问经典网络中的资源?

如果您的函数运行在某个VPC网络环境中，且需访问经典网络的资源，那么您就可以使用经典网络（ClassicLink）将资源连接到您的VPC网络中，然后授予您的函数访问该VPC的权限即可。详情请参见ClassicLink概述。

4.10.5. 在VPC中的函数如何访问公网?

函数计算在VPC功能中提供了一个启用或禁用您的函数访问公网的功能。一旦启用公网访问，函数计算将具有公网访问能力。您可以通过设置internetAccess选项启用或禁用公网访问。您也可以在您的VPC中搭建公网NAT，然后授权您在VPC中的函数访问公网的权限。

4.10.6. 遇到vSwitch is in unsupported zone的错误怎么办?

根据错误信息，找到函数计算所支持的可用区（Zone），在您的VPC中，在此可用区创建一个虚拟交换机（vSwitch），然后以此交换机配置函数计算服务的VPC。如果在此交换机中未能找到函数计算，说明您的虚拟交换机未创建成功，您需要重新在VPC环境中函数计算支持的可用区内，创建一个虚拟交换机（vSwitch），并在函数计算服务的VPC配置中设置此vSwitch ID。

配置成功后，在函数中就可以访问您的VPC资源了。同一个VPC内不同可用区的交换机默认内网互通，详细信息，请参见创建和管理专有网络。

4.10.7. 开启VPC功能对可用性的影响

如果您在华东1（杭州）、华东2（上海）、华北2（北京）和华南1（深圳）这四个地域内购买了函数计算的资源，且需要使用VPC功能，那么您可以根据函数计算控制台的提示提交使用申请。提交申请后，后台会为您进行VPC迁移。本文介绍您在迁移VPC时遇到的问题及相关解决方案。

> **说明** 您可以在函数计算控制台**概览**页面，查看开通VPC功能的申请提示。

在迁移VPC时可能对您的业务函数造成以下影响，请确定业务使用场景及相关兼容性的解决方案后再提交申请，迁移成功后不会影响您的服务性能。若您需要了解更多VPC内资源，更多信息，请参见配置网络。

- 迁移VPC过程中，您在请求以下API时遇到秒级别的503错误信息，请尽量重试。
  - CreateService
  - UpdateService
  - DeleteService
  - CreateFunction
  - UpdateFunction
  - DeleteFunction
  - CreateTrigger
  - UpdateTrigger
  - DeleteTrigger
  - GetTrigger
  - ListTriggers
- 迁移VPC后，如果您的函数存在以下情况，会影响您业务的正常运行。
  - 函数中存在依赖阿里云的服务，并且在调用这些服务时使用了内网Endpoint，如果这些服务提供的VPC Endpoint和内网Endpoint是分离的，例如表格存储、密钥管理服务和容器镜像服务，迁移后服务的内网Endpoint访问遇到网络不通时该怎么办?
    您可以采用以下方案解决:
    您可以将这些服务的访问Endpoint替换成VPC Endpoint。如果该服务没有提供VPC Endpoint例如表格存储，那么您就可以使用其公网Endpoint。

    > **说明** 关于使用公网Endpoint会产生的相应公网流量费用，请参见产品计费。

  - 迁移VPC后，函数访问我在经典网络中的云服务器的资源，并且访问时使用了云服务器的内网IP地址，影响业务的正常运行怎么办?
    您可以采用以下方案解决:
    - 方案一：将云服务器的资源迁移到VPC网络中，使用函数计算的VPC功能访问。
    - 方案二：将云服务器内网IP地址换成公网IP地址访问。

      > **说明** 关于方案二产生的相应公网流量费用，请参见产品计费。

- 迁移VPC后，函数访问我在经典网络中的云数据库服务，并且访问时使用了内网IP地址的情况，影响业务的正常运行怎么办？
  您可以采用以下方案解决：
  - 方案一：将云数据库服务迁移到VPC网络中，使用函数计算的VPC功能访问。更多信息，请参见更改网络类型。
  - 方案二：使用云数据库的公网访问方式。

    > **说明** 关于方案二会产生的相应公网流量费用，请参见产品计费。

如果您有任何反馈或疑问，请联系我们 。

5.函数管理

5.1. 管理函数

函数计算的资源调度与运行以函数为单位。函数必须从属于服务，同一个服务下可以创建多个函数，这些函数共享一些相同的设置，例如服务授权、日志配置，但彼此相互独立，互不影响。本文介绍函数的属性和运行环境，以及如何通过函数计算控制台创建、配置和删除函数。

前提条件

创建服务

支持的函数运行环境列表

运行环境	说明	文档链接
Node.js 6	Node.js 6版本	Node.js运行环境
Node.js 8	Node.js 8版本	
Node.js 10	Node.js 10版本	
Node.js 12	Node.js 12版本	
Node.js 14	Node.js 14版本	
Python 2.7	Python 2.7版本	Python运行环境
Python 3	Python 3.6版本	
Python 3.9	Python 3.9版本	
PHP 7.2	PHP 7.2.7版本	PHP运行环境
Java 11	Java 11版本	Java运行环境
Java 8	Java 8版本	
.NET Core 2.1	.NET Core 2.1版本	.NET Core运行环境
Go 1.x	Go 1.8或以上版本	Go运行环境
Custom Runtime	无	Custom Runtime运行环境
Custom Container	无	Custom Container运行环境

创建函数

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在**函数管理**页面，单击**创建函数**。
6. 在**创建函数**页面，按需选择创建函数的方式。

   本文以**使用标准 Runtime 从零创建**为例。更多创建函数的方式，请参见以下文档：

   - 使用容器镜像创建函数
   - 使用模板创建函数

7. 在**基本设置**区域，填写相关参数，然后单击**创建**。

   参数说明如下所示。

参数	是否必填	操作	示例值
函数名称	否	填写自定义的函数名称。 **说明** 如果不填写名称，函数计算会自动为您创建。	Function
运行环境	是	选择您熟悉的语言，例如Python、Java、PHP、Node.js等。函数计算支持的运行环境，请参见管理函数。	Node.js 12
代码上传方式	否	默认选择**使用示例代码**创建函数，函数创建完成后自带函数计算平台为其配置的示例代码。您也可以选择以下三种方式上传您自己的代码。 ○ **通过 ZIP 包上传代码**：选择函数代码ZIP包。 ○ **通过文件夹上传代码**：选择包含函数代码的文件夹。 ○ **通过 OSS 上传代码**：选择上传函数代码的**Bucket名称**和**文件名称**。	使用示例代码
启动命令	否	程序的启动命令。如果不配置启动命令，您需要在代码的根目录手动创建一个名称为bootstrap的启动脚本，您的程序通过此脚本来启动。 **说明** 仅当您选择**使用自定义运行时平滑迁移Web Server**时，需配置此参数。	npm run start
监听端口	是	您的代码中的HTTP Server所监听的端口。 **说明** 仅当您选择**使用自定义运行时平滑迁移Web Server**时，需配置此参数。	9000
请求处理程序类型	是	选择请求处理程序类型。 ○ **处理事件请求**：通过定时器、调用API/SDK或其他阿里云服务的触发器来触发函数执行。 ○ **处理 HTTP 请求**：用于处理HTTP请求或Websocekt请求的函数。如果您的使用场景是Web场景，建议您**使用自定义运行时平滑迁移Web Server**。具体信息，请参见函数计算控制台。	处理事件请求
实例类型	是	选择适合您的实例类型。 ○ **弹性实例** ○ **性能实例** 更多信息，请参见实例类型及使用模式。关于各种实例类型的计费详情，请参见计费概述。	弹性实例

参数	是否必填	操作	示例值
内存规格	是	设置函数执行内存。 ○ **选择输入**：在下拉列表中选择所需内存。 ○ **手动输入**：仅适用于弹性实例，单击**手动输入内存大小**，可自定义函数执行内存。取值范围[128, 3072]，单位为MB。 **说明** 输入的内存必须为64 MB的倍数。	512 MB
实例并发度	是	设置函数实例的并发度，具体信息，请参见设置实例并发数。	1
请求处理程序	是	设置请求处理程序，函数计算的运行时会加载并调用您的请求处理程序处理请求。	index.handler

说明 您也可以在**配置触发器**区域为即将创建的函数配置触发器，具体操作步骤，请参见触发器管理。

函数创建完成后，在**函数管理**页面，即可查看已创建的函数。

获取函数ARN

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在**函数管理**页面，单击目标函数名称。
6. 在函数详情页面的**函数概览**区域，复制函数ARN。

   资源ARN（Aliyun Resource Name）可用于在代码中定位阿里云资源。

更新函数

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在**函数管理**页面，单击目标函数**操作**列的**配置**。
6. 在编辑函数配置页面，按需修改相应的配置项，单击**保存**。
   - （可选）**基础信息**

     配置函数的基础信息。目前仅支持修改函数**描述**、**实例类型**、**内存规格**和**实例并发度**。

     注意 **实例类型**的修改遵循以下原则：
     - 支持**弹性实例**与**性能实例**互相修改。
     - 从**性能实例**修改为**弹性实例**时，即从高配置到低配置的降低实例规格，系统会检测切换的风险，同时内存规格会自动降配。
   - （可选）**环境信息**

     配置函数的运行环境和其他关键参数。

参数	说明

参数	说明
运行环境	选择您熟悉的语言，例如Python、Java、PHP、Node.js等。函数计算支持的运行环境，请参见支持的函数运行环境列表。
请求处理程序（函数入口）	修改请求处理程序。格式为[文件名].[函数名]。
执行超时时间	设置超时时间。默认为60秒，最长为86400秒。 **说明** 超过设置的超时时间，函数将以执行失败结束。如需更长的时间限制，请提交工单。

- （可选）**其他配置**

配置项	说明	参考文档
生命周期函数	配置函数的生命周期函数，包括**初始化函数**、**PreFreeze 函数**和**PreStop 函数**。	函数实例生命周期
层	选择已创建的层名称和版本。层可以为您提供自定义的公共依赖库、运行时环境及函数扩展等发布与部署能力。	构建层
环境变量	配置函数运行环境中的环境变量。	环境变量
DNS 配置	配置自定义DNS，平台根据该配置生成*resolv.conf*。	在函数中配置自定义DNS

说明 您也可以根据提示删除不需要的函数。

更多信息

除控制台外，函数计算还提供了以下方式来管理函数：

- Serverless Devs。
- API概览。

5.2. 为函数安装第三方依赖

函数计算运行环境中已经内置了常用的依赖供您直接在函数中引用，同时，函数计算支持使用第三方依赖。本文介绍如何安装第三方依赖。

背景信息

您可以在以下文档的内置模块章节，查看函数计算内置的常用依赖：

- Node.js环境说明
- Python环境说明
- 环境说明
- 编译部署代码包
- .NET Core运行环境
- Custom Runtime环境说明

安装方法

您可以通过以下方式安装第三方依赖：

- 通过函数计算控制台：

- 通过代码包安装依赖。
  a. 将第三方依赖与代码文件打包。

  > 注意
  > - 您需要进入代码目录，打包所有文件。打包完成后，入口函数文件需要位于包内的根目录。
  > - 不同系统下打包方式不同，请您根据实际情况选择合适的打包方式。

  b. 登录函数计算控制台，通过**上传zip包**、**上传文件夹**或**通过 OSS 上传**的方式上传代码包，部署函数。
- 通过终端安装依赖。
  当您的函数计算控制台使用新版代码在线编辑器时，您可以在**函数代码**页签的终端内执行依赖安装命令，进行代码依赖的安装。例如以Python为例，在新版代码在线编辑器的终端内执行以下命令安装依赖。

```
pip install -t . demo #demo为pip包的文件名称，执行此命令前您需将demo替换为pip包的文件名称。
```

- 使用Serverless Devs安装：通过函数计算的Serverless Devs，创建并部署函数。具体操作，请参见命令列表。

关于函数计算安装第三方依赖的总结，请参见函数计算安装依赖库方法小结。

参考信息

您可以参考以下文档的使用自定义模块章节，查看不同运行环境下的安装第三方依赖的操作步骤：

- 环境说明
- HTTP请求处理程序（HTTP Handler）
- 环境说明
- 编译部署代码包
- .NET Core运行环境
- Custom Runtime运行环境

5.3. 层管理

5.3.1. 构建层

层可以为您提供自定义的公共依赖库、运行时环境及函数扩展等发布与部署能力。您可以将函数依赖的公共库提炼到层，以减少部署、更新时的代码包体积。本文介绍层的功能原理、各运行时使用层说明以及如何构建层的ZIP包。

功能原理

- 构建层时，需要将所有内容打包到ZIP文件中。函数计算运行时会将层的内容解压并部署在*/opt*目录下。
- 当函数配置多个层时，这些层的内容将被合并至*/opt*目录，多个层按照层配置的逆序合并。如果多个层中有同名文件，先配置的层会覆盖后配置的层中的同名文件。
- 如果层中的代码依赖二进制的库或可执行文件，则需要使用Linux系统编译构建层，推荐使用Debian 9。
- 如果层中的库对指令集有依赖，请使用amd64架构的机器，或者通过交叉编译的方式确保库与函数计算运行时的兼容性。

各运行时的层文件目录与文件结构

对于支持层功能的运行时，函数计算会将特定的目录添加到运行时语言的依赖包搜索路径中，如下表所示。如果在层ZIP包中定义了与其相同的文件夹结构，则函数代码无需指定路径即可访问层。

各运行时支持添加的特定目录

运行时	特定的目录
Python	/opt/python
Node.js	/opt/nodejs/node_modules

运行时	特定的目录
Java	/opt/java/lib
PHP	/opt/php
除Custom Runtime和Custom Container之外的运行时	/opt/bin (PATH)
	/opt/lib (LD_LIBRARY_PATH)

各运行时的层ZIP包文件结构

Python Node.js Java PHP

构建层的ZIP包

创建层时，需要将所有内容打包到ZIP文件中。函数计算运行时会将层的内容解压并部署在*/opt*目录下。

构建层的ZIP包的方式和构建代码包的方式类似，为使函数在运行时能正确加载以层发布的库，库的代码目录结构需遵从各个语言标准的目录规范，具体信息，请参见各运行时的层文件目录与文件结构。对于部署于层的函数依赖库，如果按照规范的方式打包，函数计算运行时会为您自动添加各语言的依赖库搜索路径，您无需指定全路径。如您想自定义层的目录结构，需要在代码中显式添加依赖库搜索地址。具体操作步骤，请参见如何在Custom Runtime中引用层中的依赖？。

各运行时构建层ZIP包的操作步骤如下所示：

> **说明** 下面创建的工作目录*my-layer-code*仅为示例，您在实际操作过程中可以按需替换目录名称。

Python Runtime

1. 执行以下命令创建工作目录。

   ```
   mkdir my-layer-code
   ```

2. 进入已创建的工作目录。

   ```
   cd my-layer-code
   ```

3. 执行以下命令安装依赖库到*my-layer-code/python*。

   ```
   pip install --target ./python ${PackageName}
   ```

   `${PackageName}`是您要安装的依赖包的名称，`pip install`命令的详细使用方法，请参见pip install。
   代码示例如下：

   ```
   pip install --target ./python numpy
   ```

   安装成功后，文件目录结构如下所示：

   ```
   my-layer-code
   └── python
       ├── bin
       ├── numpy
       ├── numpy-1.22.4.dist-info
       └── numpy.libs
   ```

4. 在*my-layer-code*中执行以下命令打包依赖。

   ```
   zip -r my-layer-code.zip python
   ```

Node.js Runtime

1. 执行以下命令创建工作目录。

```
mkdir my-layer-code
```

2. 进入已创建的工作目录。

```
cd my-layer-code
```

3. 执行以下命令安装依赖库到*my-layer-code/nodejs*。

```
npm install --prefix ./nodejs --save ${PackageName}
```

`${PackageName}` 是您要安装的依赖包的名称，`npm install` 命令的详细使用方法，请参见npm-install。
代码示例如下：

```
npm install --prefix ./nodejs --save uuid
```

安装成功后，文件目录结构如下所示：

```
my-layer-code
└── nodejs
    ├── node_modules
    │   └── uuid
    ├── package-lock.json
    └── package.json
```

4. 在*my-layer-code*中执行以下命令打包依赖。

```
zip -r my-layer-code.zip nodejs
```

Java Runtime

1. 执行以下命令创建工作目录。

```
mkdir my-layer-code/java
```

2. 进入已创建的工作目录。

```
cd my-layer-code/java
```

3. 通过Maven安装依赖。
   i. 在*my-layer-code/java*目录下创建一个 `pom.xml` 文件。
   `pom.xml` 示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.
org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>maven.util</groupId>
    <artifactId>install-layer</artifactId>
    <version>1.0</version>
    <!-- 要下载的maven依赖 -->
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.12.0</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <!-- maven相关下载插件 -->
            <plugin>
                <artifactId>maven-dependency-plugin</artifactId>
                <configuration>
                    <!-- 是否排除间接依赖。默认false，不排除 -->
                    <excludeTransitive>false</excludeTransitive>
                    <!-- 是否消除依赖jar包后缀的版本信息。默认是false，不取消版本信息 -->
                    <stripVersion>false</stripVersion>
                    <!-- 输出文件路径 -->
                    <outputDirectory>./lib</outputDirectory>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

示例解析如下：

- 要安装的依赖包为 `org.apache.commons.lang3` 。
- 使用 `maven-dependency-plugin` 将需要安装的依赖包拷贝到*/java/lib*目录下。

ii. 在*my-layer-code/java*目录下执行以下命令安装依赖。

```
mvn dependency:copy-dependencies
```

安装完成后目录结构如下：

```
my-layer-code
└── java
    └── lib
        └── commons-lang3-3.12.0.jar
```

4. 在*my-layer-code*目录执行以下命令打包依赖。

```
zip -r my-layer-code.zip java
```

PHP Runtime

1. 执行以下命令创建工作目录。

```
mkdir -p my-layer-code/php
```

2. 进入已创建的工作目录。

```
cd my-layer-code/php
```

3. 通过Composer安装依赖。
   i. 在*my-layer-code/java*目录下创建一个 `composer.json` 文件。
   示例如下：
   ```
   {
     "require": {
       "aliyunfc/fc-php-sdk": "~1.2",
       "alibabacloud/fnf": "^1.7"
     }
   }
   ```
   ii. 执行以下命令安装依赖。
   ```
   composer install
   ```
   安装成功后目录结构如下：
   ```
   layer-code
    └─php
      ├──composer.json
      ├──composer.lock
      └──vendor
   ```
4. 在*my-layer-code*中执行以下命令打包依赖。
   ```
   zip -r my-layer-code.zip php
   ```

5.3.2. 使用层

本文介绍如何在函数中使用层，包括创建层、配置层和删除层。

通过控制台创建和配置

前提条件

- 创建函数
- 构建层

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**高级功能 > 层管理**。
3. 在顶部菜单栏，选择地域。
4. 在**层管理**页面，单击**创建层**。
5. 在**创建层**页面，设置相关参数，然后单击**创建**。

参数	描述
名称	设置层的名称，支持自定义。
描述	设置层的描述信息，便于区分层。
兼容运行时	选择创建层兼容的运行时环境。

参数	描述
层上传方式	按需选择上传代码的方式。包含以下三种上传方式： ○ **通过 ZIP 包上传层**：选择构建层的ZIP文件。 ○ **通过文件夹上传层**：选择构建层的文件所在的文件夹。 ○ **通过 OSS 上传层**：选择上传构建层文件的**Bucket 名称**和**文件名称**。 ○ **在线构建依赖层**：选择**构建环境**并上传构建层的依赖文件 `package.json` 或 `requirements.txt` 。 **说明** 目前只支持Python和Node.js。

层创建成功后，将自动生成层的版本，版本号从1开始递增。

6. （可选）创建新版本。

   说明 已创建的层或相关层版本暂不支持修改，如果您需要修改层的相关配置，可以创建新的层或创建新版本。需要注意的是，在更新层配置时，必须先删除对已删除版本的引用。

   i. 在**层管理**页面，单击目标层名称或**操作**列的**版本管理**。
   ii. 在**版本管理**区域，单击**创建版本**。
   iii. 在创建新的版本页面，选择运行环境并上传新的层代码，然后单击**创建**。

7. 为函数配置层和层版本。

   为函数配置层时，如果层有多个版本，将按照具体的层版本与函数进行绑定。

   i. 在左侧导航栏，单击**服务及函数**。
   ii. 在**服务列表**页面，单击目标服务。
   iii. 在**函数管理**页面，单击目标函数**操作**列的**配置**。
   iv. 在**层**区域单击**+添加配置**，在**层名称**和**层版本**下拉列表选择所需的配置，然后单击**保存**。

   说明 一个函数最多支持配置5个层。当函数配置多个层时，这些层的内容将被合并至*/opt*目录，多个层按照层配置的逆序合并。如果多个层中有同名文件，先配置的层会覆盖后配置的层中的同名文件。

通过Serverless Devs创建和配置

前提条件

- 安装Serverless Devs
- 配置Serverless Devs
- 构建层

操作步骤

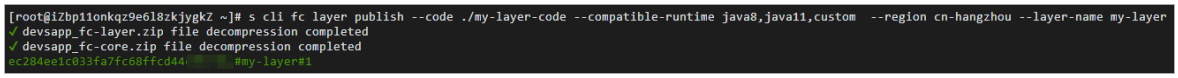
1. 执行以下命令创建层。

```
s cli fc layer publish --code ./my-layer-code --compatible-runtime java8,Java11,custom  --region cn-hangzhou --layer-name my-layer
```

   代码解析如下：

   ○ `--code` ：用于指定代码包路径。
   ○ `--compatible-runtime` ：用于指定创建层兼容的运行时环境。
   ○ `--layer-name` ：用于指定创建层的名称。

   创建完成后提示成功并返回层的ARN，ARN通过 `#` 号分为三个部分，第一部分为账号ID的标识符，第二部分为层的名称，第三部分为层的版本，如下图所示。您也可以登录函数计算控制台查看创建的层信息。



2. （可选）重复执行以下命令为已创建的层创建新版本。

```
s cli fc layer publish --code ./my-layer-code --compatible-runtime java8,Java11,custom  --region cn-hangzhou --layer-name my-layer
```

> **说明** 已创建的层或相关层版本暂不支持修改，如果您需要修改层的相关配置，可以创建新的层或创建新版本。需要注意的是，在更新层配置时，必须先删除对已删除版本的引用。

3. 在任意目录下创建*s.yaml*文件。

   示例如下：

```
edition: 1.0.0          #  命令行YAML规范版本，遵循语义化版本（Semantic Versioning）规范
name: fcDeployApp       #  项目名称
access: "default"  #  密钥别名
services:
  fc-deploy-test: #  服务名称
    component: fc  # 组件名称
    props: #  组件的属性值
      region: cn-hangzhou
      service:
        name: fctest
        description: 'test'
        internetAccess: true
      function:
        name: emoji #  函数名称
        description: this is a emoji
        runtime: java11
        codeUri: ./
        handler: index.handler
        memorySize: 128
        timeout: 6
        layers:    # 函数绑定层，取值是层的ARN
          - ec284ee1c033fa7fc68ffcd44****#my-layer#1
          - ec284ee1c033fa7fc68ffcd44****#my-layer#2
```

4. 在*s.yaml*文件所在目录执行以下命令部署函数并为其配置层。

```
s deploy
```

删除层与层版本

您可以根据提示删除不需要的层或者层版本。已删除的层不能再被查看或被函数配置引用，但已配置引用该层的函数执行不受影响。

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**高级功能 > 层管理**。
3. 在顶部菜单栏，选择地域。
4. 在**层管理**页面，按需选择删除层或层版本。
   - 删除层
     单击目标层**操作**列的**删除**，在弹出的对话框中选中确认删除层中的全部N个版本复选框，然后单击**删除**。
   - 删除层版本
     单击目标层名称，在层详情页面的**版本管理**区域，单击目标层版本**操作**列的**删除**，在弹出的**确认**对话框中单击**删除**。

更多信息

您还可以通过API/SDK来管理层和配置层。您可以在以下API中配置层：

- CreateFunction - 创建函数
- UpdateFunction - 更新函数

您可以通过OpenAPI Explorer调用API和SDK。

5.4. 环境变量

您可以使用环境变量，在不修改代码的前提下灵活调整FC函数的行为。环境变量作为函数配置的一部分，以字符串键值对的方式存储，不同函数拥有独立的环境变量。本文介绍环境变量的基本信息、控制台部署方式和示例代码等。

使用场景

- 跨平台或服务部署
  同一份代码，在测试环境和生产环境运行时的配置可能存在差异。您可以通过环境变量的方式选择使用不同的OSS的Bucket、数据库或者表格。函数在不同平台部署时无需更改代码。
- 配置密钥
  连接数据库的用户名和密码、您的阿里云AccessKey或者其他具有安全敏感性的认证信息都可以通过环境变量设置。
- 配置系统变量
  在配置PATH和HOME等目录时能够更灵活地使用系统库。

通过控制台配置环境变量

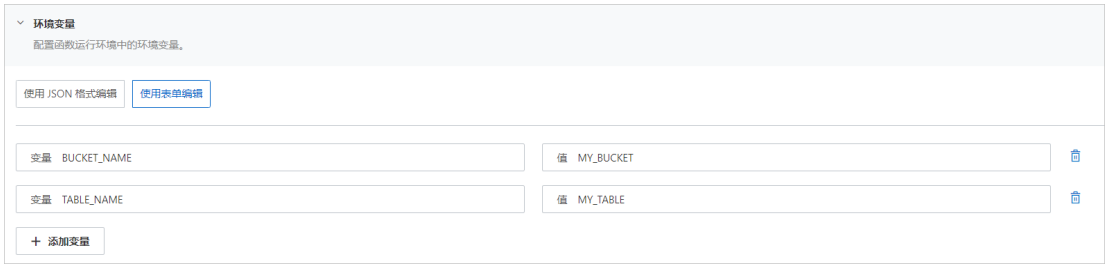
前提条件

创建函数

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。
5. 在**函数管理**页面，单击目标函数**操作**列的**配置**。
6. 在编辑函数配置页面的**环境变量**区域，按需选择配置环境变量的方式，然后单击**保存**。
   - 使用表单编辑（默认方式）
     a. 单击**+添加变量**。
     b. 配置环境变量的键值对：
        - **变量**：自定义。
        - **值**：自定义。

        示例如下：

   - 使用JSON格式编辑
     a. 单击**使用JSON格式编辑**。

b. 在文本框内，输入对应的JSON格式的键值对，格式如下：

```
{
    "key": "value"
}
```

示例如下：

```
{
    "BUCKET_NAME": "MY_BUCKET",
    "TABLE_NAME": "MY_TABLE"
}
```

成功保存后，页面会跳转至函数详情页面。

7. 验证环境变量是否创建成功。

i. 在函数详情页面，单击**函数代码**页签。

ii. 在代码编辑器中编写代码，然后单击**保存并部署**，等待部署成功后单击**测试函数**。

本文以Python事件函数为例，验证环境变量的代码示例如下：

```
# -*- coding: utf-8 -*-
import logging
import os
def handler(event, context):
    logger = logging.getLogger()
    value = os.environ.get('BUCKET_NAME')
    logger.info('BUCKET_NAME: {}'.format(value))
    value = os.environ.get('TABLE_NAME')
    logger.info('TABLE_NAME: {}'.format(value))
    return "done"
```

iii. 在**函数代码**页签，查看**日志输出**。

从打印日志中可以看出环境变量已经创建成功。

通过Serverless Devs配置环境变量

前提条件

- 安装Serverless Devs
- 配置Serverless Devs

操作步骤

1. 创建一个用于测试的代码目录，如 `test` 。
2. 进入 `test` 目录，基于上述代码创建 `index.py` 文件。
3. 创建 `s.yaml` 文件。

文件内容如下：

```
edition: 1.0.0
name: transform_fun
access: default
vars:
  region: cn-shenzhen
services:
  fc-Demo-envdemo:
    component: devsapp/fc
    props:
      region: ${vars.region}
      service:
        name: Demo     #  声明一个名称为Demo的服务
        internetAccess: true
      function:
        name: envdemo    #  在Demo服务下声明一个名称为envdemo的函数
        handler: index.handler     #  配置函数的请求处理程序
        runtime: python3           #  配置函数的运行环境
        environmentVariables:      #  为函数配置如下两个环境变量
          BUCKET_NAME: MY_BUCKET
          TABLE_NAME: MY_TABLE
        codeUri: ./                #  从当前目录"./"部署函数。部署函数时，Serverless Devs会将当前目录打包上传。
```

4. 执行 `s deploy` 部署项目。
   执行成功后，您可以登录函数计算控制台查看已创建的服务和函数，以及为该函数配置的两个环境变量。

通过SDK配置环境变量

以Python SDK为例，环境变量的参数为environmentVariables，参数取值以字典形式存储。创建、更新、获取环境变量的示例代码如下。

- 创建环境变量

```
# coding: utf-8
import fc2
client = fc2.Client(
    endpoint='your endpoint', #接入点信息，具体请参见服务地址https://help.aliyun.com/document_detail/52984.htmlhttps://www.alibabacloud.com/help/en/function-compute/latest/endpoints。
    accessKeyID='your accessKeyID', #AccessKey ID，阿里云身份验证，在RAM控制台创建。
    accessKeySecret='your accessKeySecret') #AccessKey Secret，阿里云身份验证，在RAM控制台创建。
client.create_service('test')
client.create_function(
    'test', 'test_env', 'python3',  'main.handler',
    codeDir='/path/to/code/', environmentVariables={'testKey': 'testValue'})
#test 服务名
#test_env 函数名
#python3  Runtime类型
#main.handler 请求处理程序
#codeDir 代码目录
#environmentVariables 要配置的环境变量
res = client.get_function('test', 'test_env')
print res.data
```

- 更新环境变量

```
client.update_function(
    'test', 'test_env', 'python3',  'main.handler',
    codeDir='/path/to/code/', environmentVariables={'newKey': 'newValue'})
res = client.get_function('test', 'test_env')
print res.data
```

- 获取环境变量

```
resp = client.get_function('test', 'test_env')
env = func['environmentVariables']
```

在代码中使用环境变量

假设配置的环境变量为 `{"key":"val"}` ，以下为各运行环境读取并打印此环境变量值的方法。

Node.js | Python | Java | PHP | C#

```
var value = process.env.key
console.log(value)
```

使用限制

- 字符集规则
  - Key的字符集：必须以大小写字母开头，只能包含大小写字母、数字。
  - Value的字符集：必须是可显示的ASCII字符，不能包含中文等其他字符。
- 大小限制
  所有环境变量的大小总和不能超过4 KB。
- 系统预留环境变量
  为避免系统混淆，设置环境变量时，您不能使用系统预留的环境变量FC_*、accessKeyID、accessKeySecret、securityToken和topic。
  您可以使用以下系统环境变量：
  - FC_FUNC_CODE_PATH：代码部署目录。
  - FC_FUNCTION_MEMORY_SIZE：函数内存大小，单位为MB。

安全性

创建或者更新环境变量时，函数计算会对您的环境变量使用AES256（Advanced Encryption Standard 256）标准加密存储，在初始化函数实例时，会将环境变量解密后注入到函数实例环境中。

5.5. 在函数中配置自定义DNS

自定义DNS功能适用于加速站点访问等场景，本文介绍如何在函数计算控制台为函数配置自定义DNS。

背景信息

*resolv.conf*文件为系统配置文件，您配置自定义DNS解析属性后，原有*resolv.conf*将被覆盖。关于*resolv.conf*文件的规范，您可以在Linux系统执行 `man 5 resolver` 查看，如果提示没有man page，请先根据您使用的发行版本进行安装。例如，在CentOS下可以执行 `yum install man-pages man-db` 进行安装。详细信息，请参见Linux manual page。

使用限制

自定义DNS功能仅支持函数计算官方提供的Runtime和Custom Runtime。

前提条件

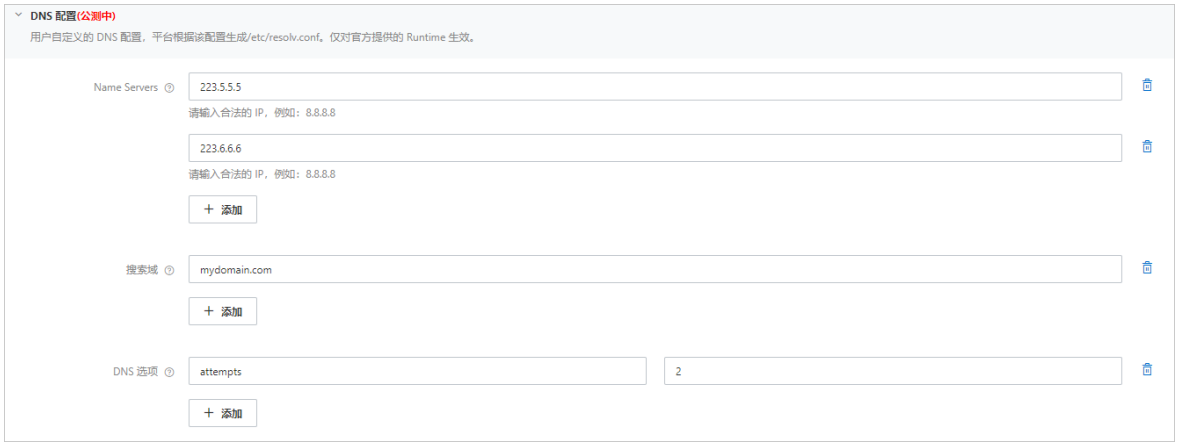
创建函数

配置DNS

> **说明** 如需使用函数计算默认的DNS服务器，请在Name Servers中添加默认IP地址*100.100.2.136*和*100.100.2.138*。

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务。

5. 在**函数管理**页面，单击目标函数**操作**列的**配置**。
6. 在编辑函数配置页面下方的**DNS 配置**区域，按需配置以下参数，然后单击**保存**。



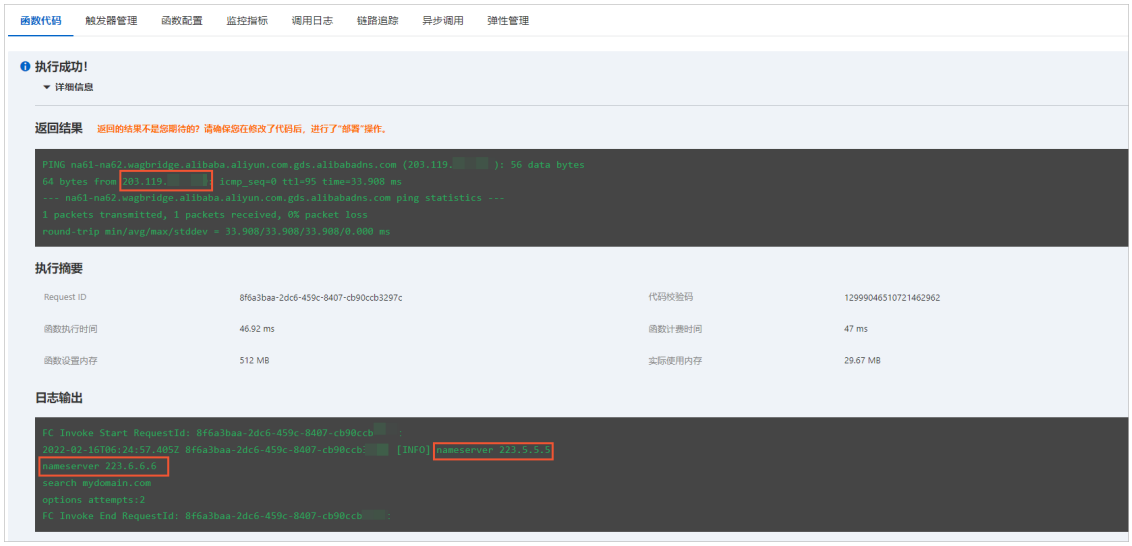
参数	说明	示例值
Name Servers	DNS服务器的IP地址。支持添加多个DNS服务器的IP地址。	○ 223.5.5.5 ○ 223.6.6.6
搜索域	DNS搜索域。支持添加多个搜索域。当访问的域名不能被DNS解析时，搜索域将被追加到无法被解析的域名后重新解析。	mydomain.com
DNS 选项	*resolv.conf*文件中的配置项。每一项对应一个键值对，格式为*key:value*，其中键为必填。	attempts:1

7. 验证配置的DNS解析属性是否生效。
   i. 在函数详情页面，单击**函数代码**页签。
   ii. 在代码编辑器中编写代码，单击**部署代码**，然后单击**测试函数**。

   代码示例（Python 3）如下：

```
# -*- coding: utf-8 -*-
import logging
import subprocess
def handler(event, context):
  logger = logging.getLogger()
  f = open('/etc/resolv.conf')
  logger.info("".join(f.readlines()))
  f.close()
  output = subprocess.check_output(["ping", "-c", "1", "www.aliyun.com"])
  return output
```

iii. 在**函数代码**页签，查看执行结果。在**返回结果**区域，查看*resolv.conf*生成的对应的内容，在**日志输出**区域，查看返回的域名的IP地址。



5.6. 函数管理FAQ

5.6.1. 函数运行FAQ

5.6.1.1. 如何监控函数执行？

函数计算每次执行都会生成执行计量数据，并将其输出，详情请参见云监控。

您可以在云监控控制台中查看函数运行情况，使用云监控来监控函数计算运行情况，详情请参见监控数据。您也可以为服务执行设置日志服务保存您的执行过程和结果，为函数计算配置日志的步骤，详情请参见配置日志。

5.6.1.2. 可执行文件在本地测试正常，但在函数计算的运行环境中执行时却返回permission denied报错，我该怎么办？

问题现象

当我在本地使用Windows操作系统或是macOS操作系统开发函数时，可执行文件在本地测试正常，但当我将函数部署到函数计算运行时，出现 `permission denied` 报错。

可能原因

由于在开发函数的过程中需要打包代码，在打包代码时，Windows操作系统和macOS操作系统中有些压缩工具在打包文件时可能会丢失文件或文件夹的属性。由于文件或文件夹的属性已丢失，文件已不具备可执行权限，代码再次在函数计算的运行环境（Linux）中解压部署运行时，就会遇到 `permission denied` 报错。

解决方案

操作步骤

1. 请确保您的压缩工具保留了文件或文件夹的Others权限，例如755权限。当您的文件或文件夹不具备该类似权限时，您可以选择以下方法解决：
   - 在函数计算的运行环境（Linux）中执行 `chmod +x 文件名` 命令赋予权限。
   - 在本地开发函数时，使用Linux操作系统开发。

5.6.1.3. 如果程序要产生很大一个文件，或者需要一个比较大的空间，在什么地方能申请到？

如果需要在内存中处理大文件：

- 适当调大内存，当前极致弹性实例最大内存是3 GB，如果有更高内存需求，可以使用极致性能实例。
- 使用流式方式处理数据，会减小内存消耗。例如对消耗1 GB内存，运行3分钟的函数进行优化，使用流式处理后，只需运行10秒，消耗256 MB内存。

如果需要在磁盘中临时存储文件：

- 函数计算为每个函数提供默认512 MB的临时存储空间，在*/tmp*目录下。
- 函数计算支持配置NAS，使用NAS对文件进行存储，挂载后操作NAS体验与操作本地磁盘相同 。

5.6.1.4. 在程序运行过程中，能从哪些接口获取到实际使用的内存？

您可以通过以下两种方式查询实际使用的内存：

- 方式一：函数调用的返回信息中包含最大内存消耗等信息。
- 方式二：在性能监控指标中可以查看连续一段时间内的函数运行资源消耗情况。详细信息，请参见监控指标。

5.6.1.5. 函数可以相互调用吗？

函数可以相互调用。在函数中调用其他函数与您在本地调用其他函数相同，需要使用InvokeFunction API调用指定函数。您可以参考各个不同语言的SDK，更多信息，请参见：

- Node.js
- Python
- Java

> **说明** 您可以从入口函数参数Context对象中获取相关的临时Token来构造对应的FC Client。

使用函数工作流编排函数，更多信息，请参见Serverless工作流。

5.6.1.6. 函数如果出现死循环，怎么杀掉函数进程？

死循环可以分为两种情况：

- 单个函数运行逻辑出现死循环。
- 多个函数调用出现无限递归，无法中止的情况。例如函数A调用B，B又调用A。

情况1：函数超时后，系统会自动中止函数，您会收到超时错误，但不会有财务损失。

情况2：您可以把被无限调用的函数的并发度设置为0，函数的调用将被终止，详情请参见函数级按量实例伸缩控制。

5.6.1.7. 如何避免循环触发？

一个典型的循环触发场景是OSS的某个Bucket上传文件触发了一个函数，这个函数处理事件，又生成了一个或多个文件，写回到OSS的Bucket里，这个写入动作又触发了函数执行，形成了链状循环。

上述过程类似于一个无限递归，为了避免这种循环触发函数，需要设定一个递归终止条件，例如将触发函数的Bucket目录前缀设置成 `src/` ，生成的文件写入的目录前缀设置为 `dst/` ，这样生成的文件就不会再次触发函数，一定要避免循环执行带来的不必要费用。

5.6.1.8. 我的执行环境是否是独立安全的？怎么保障？

您的执行环境是独立安全的，确保安全是函数计算最重要的目标。

产品和系统设计在每个方面都把安全作为最高优先级的目标。您的程序在函数计算环境中运行的隔离保护和阿里云的ECS是同一级别的，函数计算是在虚拟机级别隔离运行，并非容器。函数计算还在网络、数据、代码、防DDos攻击等方面做了完备的保护，确保您的信息安全。

5.6.1.9. 临时磁盘空间什么时候被释放？

临时磁盘空间与底层执行函数的容器生命周期一致。如果您持续有请求，那么这个容器会一直存在，因此您之前在磁盘上留下的数据也会存在。但是如果函数很长一段时间没有请求，容器被系统回收后，磁盘上的数据也会消失。

5.6.1.10. 当前目录不可写怎么办？

当您在函数中使用一些库时，可能会尝试在当前目录写入一些配置文件，此时会遇到 `[Errno 30] Read-only file system: '/code/.xxx'` 的错误，解决办法是在函数开始执行前，把进程的当前目录设置为可读写的*/tmp*目录。

> **注意** */tmp*中的数据会随着实例迭代被清空，如果您需要存储数据，您可以通过绑定NAS实现。具体信息，请参见配置NAS文件系统。

- Python

```
def my_handler(event, context):
    import os
    os.chdir('/tmp')
    // 其他代码
```

- Node.js

```
exports.handler = function(event, context, callback) {
  process.chdir('/tmp');
  // 其他代码
};
```

5.6.1.11. 我的函数会运行很久，超过函数最大Timeout时间怎么办？

函数最长运行时间为24小时，如果您的函数运行时间大于24小时，您可以将一个大函数拆分成几个小函数独立运行，使用Serverless工作流编排函数，流程中所有函数最长的运行时间为1年。

如果上述方法仍不能解决您的问题，请联系我们。

5.6.1.12. 我有10个函数，都用到了同一个公共库。我能否只上传一份公共库的代码，让这些函数共享同一份库代码？

不可以。不同的函数，负载有高有低，10000 TPS和100 TPS的函数，对资源的伸缩要求肯定不同。所以函数计算中各个函数的资源伸缩是独立进行的，函数之间的代码包是独立的，不能共享。当然，您可以将公共库存储到NAS中，不同函数启动时在NAS中加载依赖库。

5.6.1.13. 当我运行Python语言的函数时，遇到报错NoneType object has no attribute split怎么办？

可能原因

入口函数定义错误，例如您在Python事件函数的入口函数中，创建了HTTP触发器。

解决方案

操作步骤

1. 请参考以下不同的函数类型定义您的入口函数：
   - Python事件函数的入口函数定义。详细信息，请参见环境说明。

     ```
     def handler(event, context):
             return 'hello world'
     ```

   - Python HTTP函数的入口函数定义。详细信息，请参见请求处理程序（Handler）。

```
def handler(environ, start_response):
        context = environ['fc.context']
        # get request_body
        try:
            request_body_size = int(environ.get('CONTENT_LENGTH', 0))
        except (ValueError):
            request_body_size = 0
        request_body = environ['wsgi.input'].read(request_body_size)
        print 'request_body: {}'.format(request_body)
        # do something here
        status = '200 OK'
        response_headers = [('Content-type', 'text/plain')]
        start_response(status, response_headers)
        # return value must be iterable
        return [b"Hello world!\n"]
```

可能原因

HTTP函数逻辑错误，例如忘记调用 `start_response` 参数，如下所示HTTP函数的示例代码中就忘记调用此参数。

```
def handler(environ, start_response):
        # do something here
        status = '200 OK'
        response_headers = [('Content-type', 'text/plain')]
        # forget to call start_response
        # start_response(status, response_headers)
        # return value must be iterable
        return [b"Hello world!\n"]
```

解决方案

操作步骤

1. 请参考Python HTTP函数的函数入口及部署框架修改您的HTTP函数逻辑。详细信息，请参见请求处理程序（Handler）。

可能原因

在Python3的运行环境中，您将返回值设置为不可迭代的Bytes了。

解决方案

操作步骤

1. 请将返回值设置为可迭代的Bytes，例如 `[json.dumps(result).encode()]` 。

   您可以参考以下示例修改函数代码：

```
import json
def handler(environ, start_response):
        # do something here
        result = {"code": "OK"}
        status = '200 OK'
        response_headers = [('Content-type', 'application/json')]
        start_response(status, response_headers)
        # return value must be iterable
        return  [json.dumps(result).encode()]
```

5.6.1.14. 函数返回后，执行环境也会随之释放吗？我能复用上一次调用缓存的资源/状态吗？

函数运行在容器的环境里。返回后，容器不会立即释放。只有在一段时间内（取决于系统的调度算法，通常在几分钟到几十分钟），该容器都没有收到任何请求，才会被释放。如果有持续的请求，函数可以认为是常驻的。因此，您可以通过全局变量等方式缓存资源优化性能。但是，您的程序正确性不能依赖缓存一直可用这个假设，例如，当出现容器或机器宕机等情况，缓存的数据就不再有效。

5.6.2. 函数调试FAQ

5.6.2.1. 如何快速定位线上问题？

开通阿里云日志服务，然后在您的代码中添加日志，通过查询记录Request ID和日志查询快速定位问题。具体信息，请参见授予事件源访问函数计算的权限。

5.6.2.2. 为什么我的Node.js函数莫名其妙地超时了？

1. 如果使用了 `HTTP Trigger` ，确认您是否有调用 `response.send()` 。
2. 如果没有使用 `HTTP Trigger` ，确认您是否有调用入口函数 `handler` 中的 `callback` 。
3. 如果还是不确定可以通过添加日志定位超时原因。详情请参见如何快速定位线上问题。

5.6.2.3. 遇到The service or function doesn't belong to you的错误怎么办？

请检查您的endpoint中的阿里云账号ID是否正确，必须使用主账号的ID，不能是RAM子账号ID。

5.6.2.4. 遇到SDK.ServerUnreachable：SocketTimeoutException has occurred on a socket read or accept的问题怎么办？

请检查您调用Java SDK的代码，确保配置的ReadTimeoutMillis要大于您的函数超时时间。

5.6.2.5. 为什么通过浏览器访问HTTP Trigger对应的函数会强制下载？

自2018-11-19起，对于新创建的HTTP Trigger，在InvokeFunction时，服务端会为 `response header` 中强制添加 `content-disposition: attachment` 字段，此字段会使得返回结果在浏览器中以附件的方式打开。此字段无法覆盖，使用自定义域名将不受影响。

5.6.2.6. 如何确定函数内部访问其他服务的网络连通性问题？

- 如果您在VPC网络环境中访问外网连接，您可以打开VPC公网访问排除公网访问的问题。
- 您可以在函数中增加类似以下的代码并调用函数，然后在控制台观察输出，确定网络连通性及服务可用性：
  - `os.system("ping -c 5 " + host_or_ip)` 确定IP连通性问题。
  - `os.system("curl -v " + endpoint)` 验证HTTP服务是否可服务。
- 您可以在所请求的服务中查找相关连接日志，确定对应时间段内是否有来自函数计算的请求。

5.6.2.7. 如何在Custom Runtime中引用层中的依赖？

Custom Runtime是自定义运行环境，内置了多种语言版本，比如Python、Node.js、Java和PHP等，这些语言的层打包的方法和原生Runtime是一致的。但是在Custom Runtime中，您需要将层依赖所在目录添加到运行时语言的依赖包搜索路径中。本文介绍Custom Runtime内各语言引用层中依赖的方法。

示例说明

本文添加层所在目录的示例，如*/opt/python*仅为推荐的目录，如果您是按照构建层页面步骤构建的层ZIP包，则层会默认部署到该目录下。否则，请先确认您构建层ZIP包的目录结构，然后将*/opt/python*替换为实际的目录。

例如，您的层ZIP包内的目录名为*my-layer-code*，那么层会部署到*/opt/my-layer-code*目录下。您需要在函数配置中设置环境变量时，或者代码中增加依赖库搜索路径时使用*/opt/my-layer-code*路径。

在Custom Runtime-Python语言中引用层中的依赖

- 方式一：在函数配置中设置 `PYTHONPATH` 环境变量，添加层所在的目录。
  示例如下。

```
PYTHONPATH=/opt/python
```

- 方式二：在您项目的入口文件里添加以下语句，需要在导入层的依赖库前执行。

```
import sys
sys.path.append('/opt/python')
# import {PackageFromLayer}
```

具体操作步骤，请参见示例python-demo-with-lib-in-layer。

在Custom Runtime-Node.js语言中引用层中的依赖

在函数配置中设置 `NODE_PATH` 环境变量，添加层所在的目录。具体操作步骤，请参见示例nodejs-demo-with-lib-in-layer。

```
NODE_PATH=/opt/nodejs/node_modules
```

在Custom Runtime-Java语言中引用层中的依赖

- 方式一：在启动命令中设置 `-classpath` 参数，添加层所在*/opt/java/lib/**目录。

```
java -Dserver.port=9000 -classpath /code/:/opt/java/lib/* com.example.demo.DemoApplication
```

- 方式二：在函数配置中设置 `CLASSPATH` 环境变量，添加层依赖所在目录。

```
CLASSPATH=/code/:/opt/java/lib/*
```

> **注意** 使用 `CLASSPATH` 环境变量时，不支持使用参数 `-jar` 指定JAR包的方式运行程序。例如，`java -classpath ${CLASSPATH} -jar yourJarExe.jar` 示例中，使用 `-jar` 参数时，Java JVM会使用JAR包内的MANEFEST.MF，所有环境变量和命令行指定的搜索路径都会被忽略，因此指定的 `CLASSPATH` 不会生效。

在Custom Runtime-PHP语言中引用层中的依赖

在您项目的入口文件里添加以下语句，需要在导入层的依赖库前执行。

```
<?php
$path = '/opt/php';
set_include_path(get_include_path() . PATH_SEPARATOR . $path);
```

6.高级功能

6.1. 配置自定义域名

本文介绍自定义域名的典型应用场景，以及如何在函数计算控制台为Web应用绑定自定义域名，并为绑定的自定义域名设置CDN加速功能。

典型应用场景

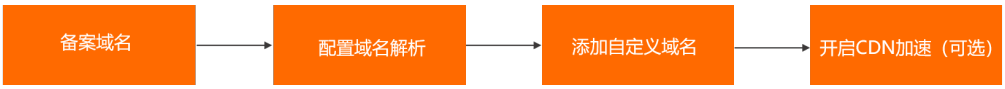
函数计算支持创建HTTP函数，且只有HTTP函数可以被HTTP请求触发执行。HTTP函数可以被看作为一个Web应用，该应用能处理HTTP请求，并将处理结果返回给调用端。在以下示例场景中，您需要为Web应用绑定自定义域名：

- 您创建了一个Web应用，并将该应用迁移到函数计算，希望用户通过固定的域名访问该应用。您可以在函数计算控制台为该Web应用绑定自定义域名，实现通过固定域名访问该应用。
- 您通过函数计算控制台搭建了一个Web应用，您可以通过函数计算提供的默认地址 `<account_id>.<region_id>.fc.aliyuncs.com/<version>/proxy/<serviceName>/<functionName>/[action?queries]` 访问该应用。在实际操作过程中，存在更改Web应用访问地址的需求，为了不影响用户的使用，您可以为Web应用绑定自定义域名，实现通过固定域名访问该应用。

前提条件

创建HTTP函数。自定义域名发出的请求只能触发HTTP函数执行。

操作流程



步骤一：备案域名

在阿里云ICP代备案管理系统备案自定义域名。具体操作，请参见ICP备案流程概述。

步骤二：配置域名解析

配置域名解析到您的函数计算对应地域的Endpoint上。具体操作，请参见配置域名解析。既可以解析到公网Endpoint，又可以解析到内网Endpoint。解析到公网Endpoint，则通过公网访问此域名；解析到内网Endpoint，则通过内网访问此域名。

> **说明** 配置域名解析到函数计算Endpoint即配置自定义域名的CNAME到函数计算Endpoint，Endpoint格式说明如下：
>
> - 内网Endpoint：格式为 `<account_id>.<region_id>-internal.fc.aliyuncs.com`，其中，`account_id` 是指您的阿里云账号（主账号）ID。例如，您的自定义域名为example.com，您的阿里云账号（主账号）ID为164901546557****，地域为华东2（上海），则内网Endpoint为 `164901546557****.cn-shanghai-internal.fc.aliyuncs.com`。
> - 公网Endpoint：格式为 `<account_id>.<region_id>.fc.aliyuncs.com`，其中，`account_id` 是指您的阿里云账号（主账号）ID。例如，您的自定义域名为example.com，您的阿里云账号（主账号）ID为164901546557****，地域为华东2（上海），则公网Endpoint为 `164901546557****.cn-shanghai.fc.aliyuncs.com`。

步骤三：添加自定义域名

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**高级功能 > 域名管理**。
3. 在顶部菜单栏，选择地域。
4. 在**域名管理**页面，单击**添加自定义域名**。
5. 在**添加自定义域名**页面，填写相关参数，然后单击**创建**。

参数	操作
域名	填写已在阿里云备案或接入备案的自定义域名名称。
HTTPS	根据需要启用或禁用HTTPS协议访问自定义域名的功能。取值如下： ○ **启用**：开启通过HTTPS协议访问自定义域名的功能。表示支持使用HTTP或HTTPS协议访问该自定义域名。 **说明** 您还可以选中**强制HTTPS**复选框，此时仅支持使用HTTPS协议访问该自定义域名，函数计算会将所有使用HTTP协议访问该自定义域名的请求重定向至HTTPS协议。 ○ **禁用**：关闭通过HTTPS协议访问自定义域名功能。表示仅支持使用HTTP协议访问该自定义域名，使用HTTPS协议将无法访问该自定义域名。
证书类型	选择要上传的证书类型，当您启用HTTPS协议访问自定义域名的功能时，需配置此参数。取值说明如下： ○ **阿里云 SSL 证书**：请选择您的阿里云SSL证书。如果**证书名称**下拉列表为空，则说明您尚未购买阿里云SSL证书，您可以登录SSL证书管理控制台购买。具体步骤，请参见购买SSL证书服务。 ○ **手动上传**：请手动输入**证书名称**，并填写**PEM 证书内容**和**PEM 证书密钥**。 **说明** 上传的证书的大小不能超过20 KB，私钥的大小不能超过4 KB。
TLS 协议版本	选择函数使用的TLS协议版本，如果不配置，则默认选择TLS 1.0及以上版本协议，包括TLS 1.0、TLS 1.1和TLS 1.2协议。取值说明如下： ○ **支持TLS 1.0及以上协议，兼容性最高，安全性较低**：表示对TLS 1.0及以上所有协议版本生效，包括TLS 1.0、TLS 1.1和TLS 1.2协议。 ○ **支持TLS 1.1及以上协议，兼容性较好，安全性较好**：表示对TLS 1.1及以上所有协议版本生效，包括TLS 1.1和TLS 1.2协议，使用TLS 1.0协议将无法访问配置的自定义域名。 ○ **支持TLS 1.2及以上协议，兼容性较好，安全性最高**：表示对TLS 1.2以上所有协议版本生效，仅包括TLS 1.2协议，使用TLS 1.0和TLS 1.1协议将无法访问配置的自定义域名。 **说明** 选择以上TLS协议版本后，您还可以选中**开启支持TLS 1.3** 复选框，表示同时支持TLS 1.3协议。

参数	操作
加密套件	选择TLS加密算法套件，如果不配置，默认选择全部加密套件。取值说明如下： ○ **全部加密套件，兼容性最高，安全性较低**：选择全部加密套件。支持以下强加密套件和弱加密套件： ■ 强加密套件 ■ TLS_RSA_WITH_AES_128_CBC_SHA ■ TLS_RSA_WITH_AES_256_CBC_SHA ■ TLS_RSA_WITH_AES_128_GCM_SHA256 ■ TLS_RSA_WITH_AES_256_GCM_SHA384 ■ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA ■ TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA ■ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA ■ TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ■ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ■ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 ■ TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ■ TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ■ TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305 ■ TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305 ■ 弱加密套件 ■ TLS_RSA_WITH_RC4_128_SHA ■ TLS_RSA_WITH_3DES_EDE_CBC_SHA ■ TLS_RSA_WITH_AES_128_CBC_SHA256 ■ TLS_ECDHE_ECDSA_WITH_RC4_128_SHA ■ TLS_ECDHE_RSA_WITH_RC4_128_SHA ■ TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA ■ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 ■ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 ○ **协议版本的自定义加密套件，请谨慎选择，避免影响业务**：选择部分支持的加密套件。下拉列表中显示所有加密套件，您可以单击加密套件右侧的图标删除安全性较弱的弱加密套件，保留您选择的TLS协议版本支持的加密套件。 关于TLS协议版本和其支持的加密套件，请参见TLS协议版本与加密套件对应关系。 函数计算对加密套件的命名使用RFC命名规范。同一个加密套件，使用不同命名规范的命名会存在差异，关于RFC和OpenSSL命名的加密套件名称差异点，请参见RFC与OpenSSL加密套件命名对照表。

参数	操作
CDN 加速	根据需要启用或禁用CDN加速开关。为域名设置CDN加速功能后，终端用户将通过CDN加速域名快速读取所需内容。 ○ **启用**：开启CDN加速功能。需要填写自定义的**CDN 加速域名**。然后登录CDN控制台为加速域名配置CNAME。更多信息，请参见步骤四：开启CDN加速（可选）。 ○ **禁用**：关闭CDN加速功能。
路由配置	设置路径与函数的对应关系，即不同的请求路径可以触发不同的函数执行。您需要设置以下字段： ○ **路径**：可以触发指定服务下的指定函数的请求路径。 ○ **服务名称**：来自指定路径的请求触发的对应服务。 ○ **函数名称**：来自指定路径的请求触发的指定服务下的对应函数。 ○ **版本或别名**：来自指定路径的请求触发的指定服务下的对应函数版本或别名。 您可以根据需要添加多个路由。更多路由信息，请参见路由匹配规则。

步骤四：开启CDN加速（可选）

为Web应用绑定自定义域名后，您可以将该自定义域名作为源站域名为其添加加速域名，然后为加速域名配置CNAME，即为域名设置CDN加速功能。将部署在函数计算的应用作为源站，将源内容发布到边缘节点，使终端用户能快速读取所需内容，有效降低访问时延，提高服务质量。更多关于CDN的信息，请参见CDN帮助文档。

> **注意** CDN加速功能会消耗公网流量，需要收取流量费用。具体信息，请参见计费概述。

方法一：通过函数计算控制台添加加速域名

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**高级功能 > 域名管理**。
3. 在顶部菜单栏，选择地域。
4. 在域名列表，找到目标域名，单击**操作**列的**编辑**。
5. 在编辑自定义域名页面，启用CDN加速功能，填写自定义**CDN 加速域名**，然后单击**保存**。

   > **说明** 一个自定义域名可以设置多个加速域名。

6. 登录CDN控制台，在**域名管理**页面的域名列表中可以看到刚创建的加速域名。

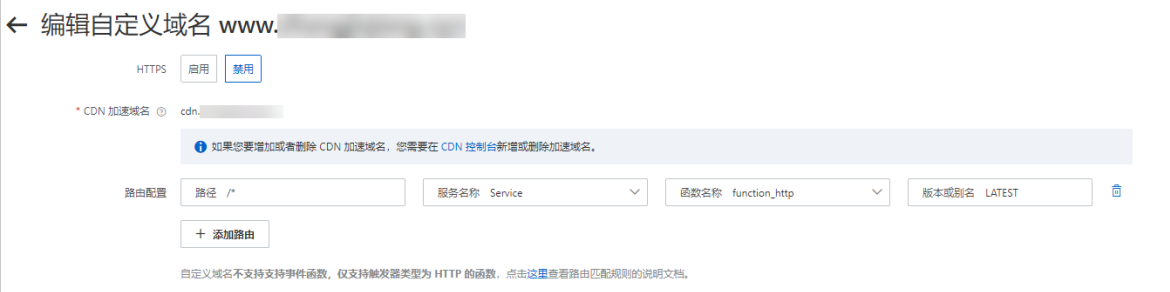
方法二：通过CDN控制台添加加速域名

通过CDN控制台开启CDN加速的详细步骤，请参见添加加速域名。

其中，设置**源站信息**时，选择**函数计算域名**，然后选择目标函数计算服务所在的地域和已创建的自定义域名。

加速域名添加成功后，您可以在函数计算控制台中的目标自定义域名中看到CDN功能开关已开启，并已绑定了在CDN控制台添加的加速域名。具体的验证操作如下：

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**高级功能 > 域名管理**。
3. 在顶部菜单栏，选择地域。
4. 在域名列表，找到目标域名，单击**操作**列的**编辑**。
5. 在编辑自定义域名页面，您可以查看由CDN控制台同步过来的CDN加速域名配置。



为自定义域名添加CDN加速域名成功后，您需要为该加速域名配置CNAME。具体操作，请参见配置CNAME。

> **说明** CNAME的格式为 `加速域名.w.alikunlun.com` ，例如 `example.aliyundoc.com.w.alikunlun.com` 。

验证结果

自定义域名或CDN加速域名设置成功后，您可以通过以下方式访问自定义域名或CDN加速域名进行测试。

- 方法一：通过命令行 `curl URL` 测试。例如 `curl example.com/login` 。
- 方法二：通过浏览器测试。
  在浏览器地址栏中输入请求URL，然后按回车键可以验证是否调用了目标函数。

TLS协议版本与加密套件对应关系

下表展示了各TLS协议版本与其支持的加密套件之间的对应关系，默认配置列为函数计算系统默认配置的加密套件。

加密套件	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	默认配置
TLS_RSA_WITH_3DES_EDE_CBC_SHA	✓	✓	✓	✗	✓
TLS_RSA_WITH_AES_128_CBC_SHA	✓	✓	✓	✗	✓
TLS_RSA_WITH_AES_256_CBC_SHA	✓	✓	✓	✗	✓
TLS_RSA_WITH_AES_128_GCM_SHA256	✗	✗	✓	✗	✓
TLS_RSA_WITH_AES_256_GCM_SHA384	✗	✗	✓	✗	✓
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	✓	✓	✓	✗	✓
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	✓	✓	✓	✗	✓
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	✓	✓	✓	✗	✓
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	✓	✓	✓	✗	✓
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	✓	✓	✓	✗	✓
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	✗	✗	✓	✗	✓
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	✗	✗	✓	✗	✓

加密套件	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	默认配置
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	✗	✗	✓	✗	✓
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	✗	✗	✓	✗	✓
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305	✗	✗	✓	✗	✓
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305	✗	✗	✓	✗	✓
TLS_RSA_WITH_RC4_128_SHA	✗	✗	✗	✗	✓
TLS_RSA_WITH_AES_128_CBC_SHA256	✗	✗	✗	✗	✓
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	✗	✗	✗	✗	✓
TLS_ECDHE_RSA_WITH_RC4_128_SHA	✗	✗	✗	✗	✓
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	✗	✗	✗	✗	✓
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	✗	✗	✗	✗	✓
TLS_AES_128_GCM_SHA256	✗	✗	✗	✓	✓
TLS_AES_256_GCM_SHA384	✗	✗	✗	✓	✓
TLS_CHACHA20_POLY1305_SHA256	✗	✗	✗	✓	✓

说明 上表中✓表示TLS协议版本支持该加密套件，✗表示TLS协议版本不支持该加密套件。

RFC与OpenSSL加密套件命名对照表

RFC命名	OpenSSL命名
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
TLS_RSA_WITH_AES_128_CBC_SHA	AES128-SHA
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE-ECDSA-AES128-SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE-ECDSA-AES256-SHA

RFC命名	OpenSSL命名
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE-RSA-DES-CBC3-SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE-RSA-AES128-SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE-RSA-AES256-SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305	不涉及
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305	不涉及
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_RSA_WITH_AES_128_CBC_SHA256	AES128-SHA256
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	ECDHE-ECDSA-RC4-SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA	ECDHE-RSA-RC4-SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256

路由匹配规则

您需要在绑定自定义域名过程中设置路径和函数的对应关系，来自不同路径的请求就可以触发不同的函数执行。函数计算支持精确匹配和模糊匹配，具体规则如下：

- 精确匹配：请求的路径和设置的路径完全一致才可以触发对应的函数。
  假设，设置路径为*/a*，对应的服务为s1，对应函数为f1，对应的版本为1。那么只有来自路径*/a*的请求才能触发版本1下的f1函数执行，来自路径*/a/*的请求无法触发版本1下的f1函数执行。
- 模糊匹配：支持使用通配符（*）设置路径，且通配符（*）只能放到路径的最后。
  假设，设置路径为*/login/**，对应服务为s2，对应函数为f2，对应版本为1。那么路径前缀为*/login/*（例如*/login/a*、*/login/b/c/d*）的请求都会触发版本1下的f2函数执行。

> **说明**
>
> - 若一个自定义域名下配置了多个路由，则精确匹配的优先级大于模糊匹配的优先级。
> - 模糊匹配时满足最长前缀匹配原则。
>   假设，配置了*/login/a/**和*/login/**两个路径，自定义域名为`example.com`，请求URL为*example.com/login/a/b*。此时，该请求URL满足设置的路径。但是根据最长前缀匹配原则，最终匹配的路径应该是*/login/a/**。

示例

假设自定义域名为`example.com`，根据本文的操作步骤，设置了以下5条路由规则。

路由规则	路径	服务名称	函数名称	版本
路由规则1	/	s1	f1	1
路由规则2	/*	s2	f2	2
路由规则3	/login	s3	f3	3
路由规则4	/login/a	s4	f4	4
路由规则5	/login/*	s5	f5	5

最终匹配结果如下。

请求URL	匹配的服务名称	匹配的函数名称	匹配的版本	匹配的路径
example.com	s1	f1	1	/
example.com/user	s2	f2	2	/*
example.com/login	s3	f3	3	/login
example.com/login/a	s4	f4	4	/login/a
example.com/login/a/b	s5	f5	5	/login/*
example.com/login/b	s5	f5	5	/login/*

问题诊断

在绑定自定义域名过程中如果发生错误，服务端会返回错误信息，下表列出了常见的错误码，帮助您快速定位和解决问题。

错误码	HTTP状态码	错误信息	原因分析
DomainNameAlreadyExists	409	domain name '%s' already exists	创建域名时，域名已存在。
DomainNameNotFound	404	domain name '%s' does not exist	获取域名信息时，域名不存在。
InvalidICPLicense	400	domain name '%s' has not got ICP license, or the ICP license does not belong to Aliyun	域名未备案，或备案未接入阿里云。
DomainNameNotResolved	400	domain name '%s' has not been resolved to your FC endpoint, the expected endpoint is '%s'	域名未设置CNAME到指定的Endpoint，可以通过**dig**命令或在域名解析服务器处查看确认。
DomainRouteNotFound	404	no route found in domain '%s' for path '%s'	没有为指定路径设置对应触发的函数。
TriggerNotFound	404	trigger 'http' does not exist in service '%s' and function '%s'	自定义域名触发的函数未设置HTTP触发器。

如果问题仍未能解决，请联系我们。

6.2. 操作审计

操作审计ActionTrail提供了行为操作日志，您可以通过操作日志对您的行为进行追踪、查看及分析。本文介绍如何在函数计算控制台查询您的操作日志。

背景信息

操作审计ActionTrail帮助您监控并记录阿里云账号的活动，包括通过阿里云控制台、OpenAPI及开发者工具对云产品的访问及管理。您可以将这些行为事件下载或保存到日志服务SLS或对象存储OSS存储空间中，然后进行行为分析、安全分析、资源变更行为追踪及行为合规性审计等操作。

操作步骤

1. 登录函数计算控制台。
2. 在左侧导航栏，选择**更多功能 > 审计日志**。
3. 在顶部菜单栏，选择地域。
4. 在**审计日志**页面，设置相关参数。
   i. 选择需要查询的事件类型。
   ii. 设置需要查询的事件时间。
   iii. 单击图标，完成搜索。

   > 注意
   > - 函数计算的操作审计日志记录了OpenAPI相关的调用事件，既可以查询通过SDK、CLI发起的OpenAPI调用记录也可以查询通过函数计算控制台产生的OpenAPI记录。操作审计支持查询的API资源类型包含函数、服务、触发器、自定义域名。更多信息，请参见API参考。
   > - 函数计算目前不支持查询API中invokeFunction的操作记录。

5. 选择目标日志，然后单击图标查看事件的相关解读。同时，您也可以在**事件解读**对话框中，单击**事件详情**查看日志的详细信息。

7.访问其他服务

7.1. 访问NAS示例

函数计算的服务配置NAS挂载点后，您可以通过编写代码访问NAS中的文件，就像访问本地文件系统一样。本文提供写入和读取NAS文件的函数代码示例。

前提条件

- 配置NAS文件系统
- 创建函数

创建写入NAS文件的函数

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务**操作**列的**函数管理**。
5. 在**函数管理**页面，单击目标函数名称。
6. 在函数详情页面，单击**函数代码**页签，在代码编辑器中编写代码。

   本文以Python 3为例，代码示例如下。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import random
import subprocess
import string
import os
def handler(event, context):
  # report file system disk space usage and check NAS mount target
  out, err = subprocess.Popen(['df','-h'], stdout = subprocess.PIPE).communicate()
  print('disk: ' + str(out))
  lines = [ l.decode() for l in out.splitlines() if str(l).find(':') != -1 ]
  nas_dirs = [ x.split()[-1] for x in lines ]
  print('uid : ' + str(os.geteuid()))
  print('gid : ' + str(os.getgid()))
  for nas_dir in nas_dirs:
    sub_dir = randomString(16)
    file_name = randomString(6)+'.txt'
    new_dir = nas_dir + '/' + sub_dir + '/'
    print('test file: ' + new_dir + file_name)
    content = "NAS here I come"
    os.mkdir(new_dir)
    fw = open(new_dir + file_name, "w+")
    fw.write(content)
    fw.close()
    # Showing the folder tree in NAS
    for root, dirs, files in os.walk(nas_dir):
      level = root.replace(nas_dir, '').count(os.sep)
      indent = ' ' * 2 * (level)
      print('{}{}/'.format(indent, os.path.basename(root)))
      subindent = ' ' * 2 * (level + 1)
      for f in files:
        print('{}{}'.format(subindent, f))
  return 'success'
def randomString(n):
  return ''.join(random.SystemRandom().choice(string.ascii_uppercase + string.digits) for _ in range
(n))
```

创建读取NAS文件的函数

1. 登录函数计算控制台。
2. 在左侧导航栏，单击**服务及函数**。
3. 在顶部菜单栏，选择地域。
4. 在**服务列表**页面，单击目标服务**操作**列的**函数管理**。
5. 在**函数管理**页面，单击目标函数名称。
6. 在函数详情页面，单击**函数代码**页签，在代码编辑器中编写代码。

   本文以Python 3为例，代码示例如下。

```
# -*- coding: utf-8 -*-
def handler(event, context):
    f = open("/mnt/test/test.txt", "r")
    print(f.readline())
    f.close()
    return 'ok'
```

   该函数的执行结果就是通过创建写入NAS文件的函数写入的内容。

7.2. 访问数据库

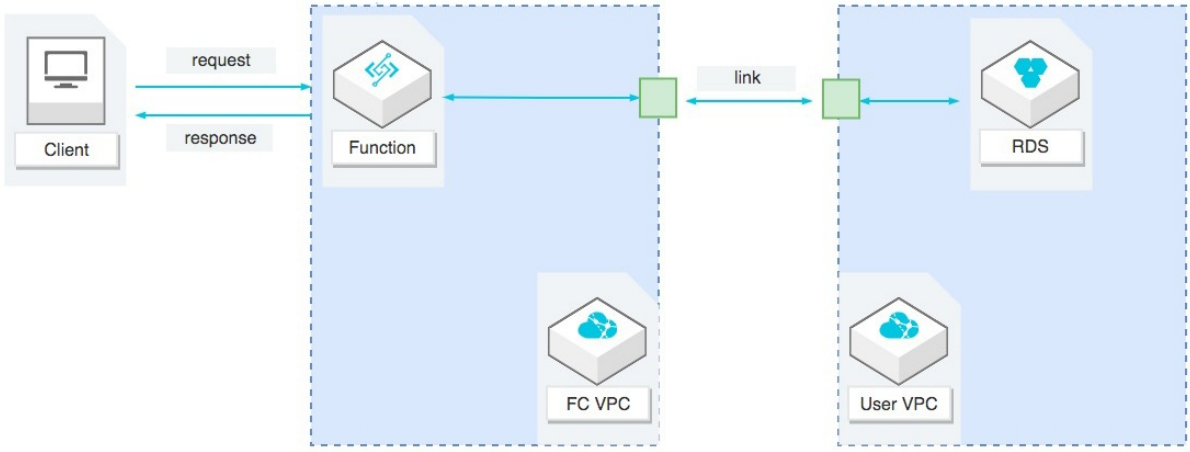
本文介绍函数计算访问VPC内数据库的机制，并提供访问数据库的示例。

访问机制

在函数计算中，运行函数的实例由函数计算动态分配，IP地址不固定，因此无法通过添加函数实例的IP地址到数据库白名单的方式来访问数据库。此外，基于最小权限原则和访问安全考虑，不建议您在生产环境中将所有IP地址（0.0.0.0/0）设置到数据库白名单中。

为了解决上述难题，您可以将数据库放置于安全的VPC中，为函数所在的服务配置允许访问该VPC内资源的功能，然后将在函数计算中配置的交换机网段添加至数据库访问白名单，即可通过VPC安全地访问您的云数据库。

函数计算访问数据库工作流程如下图所示。



1. 客户端向函数计算发出请求。
2. 函数计算根据已配置的网络设置，访问指定VPC内的数据库。
   如需通过函数计算控制台配置网络，具体步骤，请参见配置网络。您也可以使用本文中的示例方法，通过Serverless Devs的YAML文件配置。
3. 获取数据后由函数计算返回给客户端。

访问示例





7.3. 访问RDS MySQL示例

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问RDS MySQL数据库。

前提条件

- 创建RDS MySQL实例

  > **注意**
  > - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
  > - 建议您在函数计算支持的可用区创建数据库实例。
  >
  > 如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
  >
  > 函数计算支持的可用区 ❯

- 创建数据库和账号
- 在创建的数据库中，创建名为users的表。
  本文示例中，`s init`初始化的模板代码中的代码逻辑为查询名为users的数据库表中的所有数据，因此，需提前创建该表。
- 安装Serverless Devs和Docker
- 配置Serverless Devs
- 创建安全组
  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库访问IP地址白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

1. 登录RDS控制台。
2. 在左侧导航栏，单击**实例列表**。
3. 在顶部菜单栏，选择地域。
4. 在数据库实例列表页面，找到需访问的目标实例，并单击实例ID。
5. 在左侧导航栏，单击**数据安全性**。
6. 在默认显示的**白名单设置**页签，找到**default**白名单分组，在其右侧，单击**修改**。
7. 在**修改白名单分组**对话框，修改**组内白名单**，单击**确定**。

> **注意** 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP段。

部署并调试函数

1. 执行以下命令初始化项目。

```
s init start-fc-mysql-python -d start-fc-mysql-python
```

2. 执行以下命令进入项目。

```
cd start-fc-mysql-python
```

3. 编辑*s.yaml*文件。

   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-mysql-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1j6knvj40yjn0g**** # 安全组ID。
          vswitchIds:
            - vsw-bp1lgecrqfslgji9b**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: mysql
        description: visit mysql
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 60
        environmentVariables:
          MYSQL_HOST: rm-bp13d39n0m2s0****.mysql.rds.aliyuncs.com # 数据库实例的私网地址。
          MYSQL_PORT: 33** # 数据库实例的私网端口。
          MYSQL_USER: ni**** # 数据库实例的用户。
          MYSQL_PASSWORD: Hangzhou**** # 数据库实例的密码。
          MYSQL_DBNAME: django-**** # 数据库实例中创建的数据库。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-10T14:35:27.747] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Invoke Start RequestId: b00ec6c1-1d12-4110-8299-bb5129a4fd0c
2021-09-10T06:35:37.944Z b00ec6c1-1d12-4110-8299-bb5129a4fd0c [INFO] ('3', 'wan')
FC Invoke End RequestId: b00ec6c1-1d12-4110-8299-bb5129a4fd0c
Duration: 148.46 ms, Billed Duration: 149 ms, Memory Size: 256 MB, Max Memory Used: 32.21 MB
========= FC invoke Logs end =========
FC Invoke Result:
[
    "3",
    "wan"
]
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.4. 访问MongoDB示例

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问MongoDB数据库。

前提条件

- 创建适合业务需求的云数据库MongoDB实例：
  - 创建单节点实例
  - 创建副本集实例
  - 创建分片集群实例
    本文以分片集群实例为例。

> **注意**
>
> - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
> - 建议您在函数计算支持的可用区创建数据库实例。
>   如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
>
> 函数计算支持的可用区 ❯

- 创建名为test-db的数据库和名为fc_col的数据库集合。
  本文示例中，`s init`初始化的模板代码中的代码逻辑为向名为test-db的数据库的fc-col集合中插入一条文档数据；其中，test-db为环境变量*MONGO_DATABASE*的取值。
- 安装Serverless Devs和Docker

- 配置Serverless Devs
- 创建安全组
  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库访问IP地址白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

> **说明** 本文以分片集群实例为例。添加IP地址白名单的详细步骤，请参见设置白名单。

1. 登录云数据库MongoDB。
2. 在顶部菜单栏，选择地域。
3. 在左侧导航栏，单击**分片集群实例列表**。
4. 在**分片集群实例列表**页面，找到目标实例，并单击其名称。
5. 在左侧导航栏，选择**数据安全性 > 白名单设置**。
6. 在**白名单设置**页面，找到白名单分组名，在其**操作**列，单击图标，在下拉菜单中，选择**手动修改**。
7. 在**手动修改**面板，按需填写IP地址，单击**确定**。

> **注意** 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP段。

部署并调试函数

1. 执行以下命令初始化项目。

```
s init start-fc-mongodb-python -d start-fc-mongodb-python
```

2. 执行以下命令进入项目。

```
cd start-fc-mongodb-python
```

3. 编辑*s.yaml*文件。

   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-mongodb-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1j6knvj40yjn0g**** # 安全组ID。
          vswitchIds:
            - vsw-bp1lgecrqfslgji9b**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: mongodb
        description: visit mongodb
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 60
        environmentVariables:
          MONGO_DATABASE: test-db # 数据库名称。
          MONGO_URL: mongodb://m****:Hang****@s-bp1a2d6c391e****.mongodb.rds.aliyuncs.com:37** # 数据库接入点。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-22T22:25:58.139] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Invoke Start RequestId: b361814a-0a70-4511-8212-6739389d3ca4
find documents:{'_id': ObjectId('614b3ccea90d5fbfc262e918'), 'DEMO': 'FC', 'MSG': 'Hello FunctionCompute For MongoDB'}
FC Invoke End RequestId: b361814a-0a70-4511-8212-6739389d3ca4
Duration: 61.27 ms, Billed Duration: 62 ms, Memory Size: 256 MB, Max Memory Used: 51.21 MB
========= FC invoke Logs end =========
FC Invoke Result:
{'_id': ObjectId('614b3ccea90d5fbfc262e918'), 'DEMO': 'FC', 'MSG': 'Hello FunctionCompute For MongoDB'}
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.5. 访问Redis示例

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问Redis数据库。

前提条件

- 创建Redis实例

  > **注意**
  > - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
  > - 建议您在函数计算支持的可用区创建数据库实例。
  >
  > 如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
  >
  > 函数计算支持的可用区 ❯

- 安装Serverless Devs和Docker
- 配置Serverless Devs
- 创建安全组

  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库访问IP地址白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

1. 登录Redis管理控制台。
2. 在顶部菜单栏的左上角，选择实例所属的地域。
3. 在**实例列表**页，单击目标实例ID。
4. 在左侧导航栏，单击**白名单设置**。
5. 找到目标白名单分组，单击**修改**。

   > **注意** 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP地址段。

6. 单击**确定**。

部署并调用函数

1. 执行以下命令初始化项目。

   ```
   s init start-fc-redis-python -d start-fc-redis-python
   ```

2. 执行以下命令进入项目。

   ```
   cd start-fc-redis-python
   ```

3. 编辑*s.yaml*文件。

   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-redis-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1gg2tyyzuvz8x0**** # 安全组ID。
          vswitchIds:
            - vsw-bp1sciqyju9qxocys**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: redis
        description: visit redis
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 30
        initializationTimeout: 60
        initializer: index.initializer
        environmentVariables:
          REDIS_HOST: r-bp1v04cjj0jlr7****.redis.rds.aliyuncs.com # 数据库实例的私网地址。
          REDIS_PASSWORD: Hangzhou**** # 数据库实例的密码。
          REDIS_PORT: 63** # 数据库实例的私网端口。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-14T17:08:50.875] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Initialize Start RequestId: ccd73383-048d-4c8d-834e-93da59b86a21
FC Initialize End RequestId: ccd73383-048d-4c8d-834e-93da59b86a21
FC Invoke Start RequestId: eccafc0a-493e-4f3e-9afa-45c0b84a2c0f
counter: 0
FC Invoke End RequestId: eccafc0a-493e-4f3e-9afa-45c0b84a2c0f
Duration: 27.51 ms, Billed Duration: 28 ms, Memory Size: 256 MB, Max Memory Used: 34.05 MB
========= FC invoke Logs end =========
FC Invoke Result:
0
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.6. 访问RDS SQL Server示例

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问RDS SQL Server数据库。

前提条件

- 创建RDS SQL Server实例

  > 注意
  > - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
  > - 建议您在函数计算支持的可用区创建数据库实例。
  >
  > 如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
  >
  > 函数计算支持的可用区 ❯

- 创建名为users的表。
  本文示例中，`s init`初始化的模板代码中的代码逻辑为查询名为users的数据库表中的所有数据，因此，需提前创建该表。
- 安装Serverless Devs和Docker
- 配置Serverless Devs
- 创建安全组
  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库IP访问白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

1. 登录RDS控制台。
2. 在左侧导航栏，单击**实例列表**。
3. 在顶部菜单栏，选择地域。
4. 在数据库实例列表页面，找到需访问的目标实例，并单击实例ID。
5. 在左侧导航栏，单击**数据安全性**。
6. 在默认显示的**白名单设置**页签，找到**default**白名单分组，在其右侧，单击**修改**。
7. 在**修改白名单分组**对话框，修改**组内白名单**，单击**确定**。

   > 注意 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP段。

部署并调用函数

1. 执行以下命令初始化项目。

   ```
   s init start-fc-sql-server-python -d start-fc-sql-server-python
   ```

2. 执行以下命令进入项目。

   ```
   cd start-fc-postgresql-python
   ```

3. 编辑*s.yaml*文件。
   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-sql-server-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1j6knvj40yjn0g**** # 安全组ID。
          vswitchIds:
            - vsw-bp1lgecrqfslgji9b**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: sql_server
        description: visit sql server
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 60
        environmentVariables:
          MSSQL_SERVER: rm-bp1lbgwj1pw87****.sqlserver.rds.aliyuncs.com # 数据库接入点。
          MSSQL_PORT: 14** # 数据库端口。
          MSSQL_USER: sqlse**** # 数据库用户名称。
          MSSQL_PASSWORD: Hang**** # 数据库用户密码。
          MSSQL_DATABASE: ni**** # 数据库名称。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-22T22:43:24.266] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Invoke Start RequestId: 2768263f-1fb4-4146-be78-a00a6d8e656e
2021-09-22T14:43:24.816Z 2768263f-1fb4-4146-be78-a00a6d8e656e [INFO] None
FC Invoke End RequestId: 2768263f-1fb4-4146-be78-a00a6d8e656e
Duration: 46.54 ms, Billed Duration: 47 ms, Memory Size: 256 MB, Max Memory Used: 38.30 MB
========= FC invoke Logs end =========
FC Invoke Result:
null
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.7. 访问RDS PostgreSQL数据库

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问RDS PostgreSQL数据库。

前提条件

- 创建RDS PostgreSQL实例

  > **注意**
  > - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
  > - 建议您在函数计算支持的可用区创建数据库实例。
  >
  > 如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
  >
  > 函数计算支持的可用区 ❯

- 创建名为COMPANY的表，并插入ID、NAME、AGE、ADDRESS和SALARY列。
  本示例中，`s init` 初始化的模板代码中的代码逻辑为向名为COMPANY的数据库表插入一条数据，表的Schema包含ID、NAME、AGE、ADDRESS和SALARY列。因此，需提前创建该表与相应列。
- 安装Serverless Devs和Docker
- 配置Serverless Devs
- 创建安全组
  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库IP访问白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

1. 登录RDS控制台。
2. 在左侧导航栏，单击**实例列表**。
3. 在顶部菜单栏，选择地域。
4. 在数据库实例列表页面，找到需访问的目标实例，并单击实例ID。
5. 在左侧导航栏，单击**数据安全性**。
6. 在默认显示的**白名单设置**页签，找到**default**白名单分组，在其右侧，单击**修改**。
7. 在**修改白名单分组**对话框，修改**组内白名单**，单击**确定**。

   > **注意** 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP段。

部署并调用函数

1. 执行以下命令初始化项目。

   ```
   s init start-fc-postgresql-python -d start-fc-postgresql-python
   ```

2. 执行以下命令进入项目。

   ```
   cd start-fc-postgresql-python
   ```

3. 编辑*s.yaml*文件。

   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-postgresql-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1j6knvj40yjn0g**** # 安全组ID。
          vswitchIds:
            - vsw-bp1lgecrqfslgji9b**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: postgresql
        description: visit postgresql
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 30
        environmentVariables:
          DATABASE: post**** # 数据库名称。
          HOST: pgm-bp1n6g63x0xdb7oy16****.pg.rds.aliyuncs.com # 数据库接入点。
          PORT: 19** # 数据库端口。
          USER: ni**** # 数据库用户名。
          PASSWORD: Hang**** # 数据库用户密码。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-23T11:03:17.792] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Invoke Start RequestId: 7e6657d4-504a-4faa-8f59-5464df73aa65
FC Invoke End RequestId: 7e6657d4-504a-4faa-8f59-5464df73aa65
Duration: 101.87 ms, Billed Duration: 102 ms, Memory Size: 256 MB, Max Memory Used: 35.08 MB
========= FC invoke Logs end =========
FC Invoke Result:
successfully
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.8. 访问Lindorm数据库

访问数据库是指在函数计算中通过函数调用数据库应用编程接口，对数据库执行数据插入、查询等操作。通常函数计算中不同的执行环境实例之间的状态是不共享的，通过数据库可以将结构化的数据持久化以实现状态共享。本文以Python 3为例，介绍如何使用Serverless Devs部署函数来访问Lindorm数据库。

前提条件

- 创建Lindorm实例

  > **注意**
  > - 请确保您所创建的数据库实例与需要访问该数据库实例的函数在同一地域。
  > - 建议您在函数计算支持的可用区创建数据库实例。
  >
  > 如果您的资源所在的可用区不在以下列表中，可以通过在您的VPC环境中创建一个与函数计算相同可用区的vSwitch，并在函数计算的服务的VPC配置中设置此vSwitch ID。由于同一VPC内不同vSwitch之间私网互通，因此函数计算可以通过该vSwitch访问在其他可用区的VPC内的资源。具体步骤，请参见遇到vSwitch is in unsupported zone的错误怎么办？。
  >
  > 函数计算支持的可用区》

- 安装Serverless Devs和Docker
- 配置Serverless Devs
- 创建安全组
  在数据库部署的VPC内创建一个默认的安全组。如果使用已有的安全组，请确保该安全组的出口方向已放行数据库的端口和私网IP段（一般来说都是默认允许所有访问）。

配置数据库访问IP地址白名单

请使用IP地址白名单来实现安全访问，请勿使用安全组。否则，可能会导致函数连接不稳定的情况，影响业务正常运行。

1.
2.
3. 在左侧导航栏，单击**访问控制**。
4. 单击**修改分组白名单**。
5. 在**修改分组白名单**对话框，输入允许访问的IP地址段，单击**确认**。

   > **注意** 此处填写的IP地址，需为在函数计算服务的网络配置中添加的vSwitch的IP地址段。

部署并调用函数

1. 执行以下命令初始化项目。

   ```
   s init start-fc-lindorm-python -d start-fc-lindorm-python
   ```

2. 执行以下命令进入项目。

   ```
   cd start-fc-lindorm-python
   ```

3. 编辑*s.yaml*文件。

   示例如下：

```
edition: 1.0.0
name: fcDeployApp
access: "default"
services:
  fc-db-lindorm-python:
    component: devsapp/fc
    props:
      region: cn-hangzhou
      service:
        name: fc-db-demo
        description: 'demo for fc visit db'
        internetAccess: true
        vpcConfig:
          vpcId: vpc-bp1hcg467ekqsv0zr**** # 数据库实例所在的VPC ID。
          securityGroupId: sg-bp1gg2tyyzuvz8x0**** # 安全组ID。
          vswitchIds:
            - vsw-bp1sciqyju9qxocys**** # 请确保该vSwitch对应的网段已配置到数据库实例访问白名单中。
      function:
        name: lindorm
        description: visit lindorm
        runtime: python3
        codeUri: ./code
        handler: index.handler
        memorySize: 256
        timeout: 30
        environmentVariables:
          DATABASE_ADDR: ld-bp13c5s05n1mu****-proxy-lindorm.lindorm.rds.aliyuncs.com:30** # 数据库接入点。
          USER: r*** # 数据库用户。
          PASSWORD: r*** # 数据库用户密码。
```

4. 执行以下命令构建项目。

```
s build --use-docker
```

5. 执行以下命令部署项目。

```
s deploy -y
```

6. 执行以下命令调用函数。

```
s invoke -e "{}"
```

预期输出如下：

```
[2021-09-22T22:54:19.851] [INFO ] [S-CLI] - Start ...
========= FC invoke Logs begin =========
FC Invoke Start RequestId: a41a7647-cf8b-466d-a63f-59635043453e
2021-09-22T14:54:20.289Z a41a7647-cf8b-466d-a63f-59635043453e [INFO] start to test Lindorm
2021-09-22T14:54:20.291Z a41a7647-cf8b-466d-a63f-59635043453e [WARNING] Cluster.__init__ called with
contact_points specified, but no load_balancing_policy. In the next major version, this will raise a
n error; please specify a load-balancing policy. (contact_points = ['ld-bp13c5s05n1mu****-proxy-lind
orm.lindorm.rds.aliyuncs.com'], lbp = None)
2021-09-22T14:54:20.301Z a41a7647-cf8b-466d-a63f-59635043453e [WARNING] Downgrading core protocol ve
rsion from 66 to 65 for 192.168.**.**:90**. To avoid this, it is best practice to explicitly set Clu
ster(protocol_version) to the version supported by your cluster. http://datastax.github.io/python-dr
iver/api/cassandra/cluster.html#cassandra.cluster.Cluster.protocol_version
2021-09-22T14:54:20.309Z a41a7647-cf8b-466d-a63f-59635043453e [WARNING] Downgrading core protocol ve
rsion from 65 to 5 for 192.168.**.**:90**. To avoid this, it is best practice to explicitly set Clus
ter(protocol_version) to the version supported by your cluster. http://datastax.github.io/python-dri
ver/api/cassandra/cluster.html#cassandra.cluster.Cluster.protocol_version
2021-09-22T14:54:20.317Z a41a7647-cf8b-466d-a63f-59635043453e [ERROR] Closing connection <LibevConne
ction(13962401490****) 192.168.**.**:90**> due to protocol error: Error from server: code=000a [Prot
ocol error] message="Beta version of the protocol used (5/v5-beta), but USE_BETA flag is unset"
2021-09-22T14:54:20.318Z a41a7647-cf8b-466d-a63f-59635043453e [WARNING] Downgrading core protocol ve
rsion from 5 to 4 for 192.168.**.**:90**. To avoid this, it is best practice to explicitly set Clust
er(protocol_version) to the version supported by your cluster. http://datastax.github.io/python-driv
er/api/cassandra/cluster.html#cassandra.cluster.Cluster.protocol_version
2021-09-22T14:54:20.364Z a41a7647-cf8b-466d-a63f-59635043453e [INFO] Using datacenter 'datacenter1'
for DCAwareRoundRobinPolicy (via host '192.168.**.**:90**'); if incorrect, please specify a local_dc
to the constructor, or limit contact points to local cluster nodes
2021-09-22T14:54:20.550Z a41a7647-cf8b-466d-a63f-59635043453e [INFO] # row: Row(id=1, address='hangz
hou', age=11, name='testname')
FC Invoke End RequestId: a41a7647-cf8b-466d-a63f-59635043453e
Duration: 267.05 ms, Billed Duration: 268 ms, Memory Size: 256 MB, Max Memory Used: 87.87 MB
========= FC invoke Logs end =========
FC Invoke Result:
null
End of method: invoke
```

更多信息

函数计算访问数据库案例集

7.9. 访问其他服务FAQ

7.9.1. 函数计算如何访问RDS?

由于函数运行时的IP是不固定的，您需要设置RDS允许所有IP访问。但是这样会有风险，不建议这样做。函数计算支持VPC功能，您可以通过授权的方式安全地访问VPC中的资源，详情请参见配置网络。

7.9.2. 为什么我在函数里无法通过内网Endpoint访问OSS等云服务?

要在函数里通过内网Endpoint访问云服务，您需要确保函数和要访问的云服务在同一个地域内。

7.9.3. 在函数中如何访问其他服务?

函数计算只提供一个执行环境，在函数中访问其他服务例如OSS、MNS、数据库、自建的HTTP服务等，或者访问函数计算（例如调用触发一个函数），与您在本地访问这些服务是没有差异的，您只需将相应的依赖库跟函数代码一起打包上传，就可以在函数中访问相应的服务。依赖打包详情请参见为函数安装第三方依赖。

8.Serverless Devs

8.1. 从Funcraft迁移到Serverless Devs

由于后期将不再维护Funcraft，如果您是通过Funcraft实现资源管理的，为了给您带来更好的使用体验，建议将资源迁移至Serverless Devs管理。Serverless Devs具有更完善的功能例如端云联调、压测和安装依赖等。本文介绍将资源从Funcraft迁移至Serverless Devs不同组件的方式。

迁移方式

函数计算的开发团队基于Serverless Devs开发各种不同的组件，例如FC组件、ROS组件等。下文分别介绍如何从Funcraft迁移到FC组件和ROS组件，请按需选择迁移方式。关于Serverless Devs的各种组件的详细信息，请参见组件。

从Funcraft迁移到Serverless Devs的FC组件

您可以通过以下方式迁移：

- （推荐）方式一：一键式切换命令
  在Funcraft管理的函数资源目录下，执行以下命令，将Funcraft的YAML文件转换成Serverless Devs可以识别的YAML文件：

  ```
  s cli fc fun2s --target s.yaml
  ```

  成功执行一键式切换命令后，将在项目目录中生成一个*s.yaml*文件。
  在使用一键式切换命令实现迁移时，按需选择以下参数：
  - --source：指定使用Funcraft管理的函数资源路径。
  - --target：指定迁移后的YAML文件名称。默认迁移后的文件名称是*s.yaml*或*s.yml*，如果您想自定义YAML文件的名称，则需要使用此参数指定。
  - --force：如果在迁移时出现一样的文件名称，可以通过该参数强制覆盖已存在的同名称文件。
  - --region string：指定部署资源的地域。

  输出示例：

  ```
   fc-transform.zip file decompression completed
  [2021-09-02T10:31:28.870] [INFO ] [FC-TRANSFORM] - Using funcraft yaml: /test/demo/template.yml
  [2021-09-02T10:31:28.884] [INFO ] [FC-TRANSFORM] - Reminder serverless devs yaml path: /test/demo/s.y
  aml
  Tips for next step
  ======================
  * Invoke Event Function: s local invoke -t s.yaml
  * Invoke Http Function: s local start -t s.yaml
  * Deploy Resources: s deploy -t s.yaml
  End of method: fun2s
  ```

- 方式二：资源同步
  当您的资源已成功部署到函数计算时，您可以使用同步管理资源的相关命令，进行资源同步，从而实现使用Serverless Devs管理项目。关于如何同步管理资源，请参见同步资源。

从Funcraft迁移到Serverless Devs的ROS组件

在Funcraft管理的函数资源目录下，执行以下命令，将Funcraft的YAML文件转换成Serverless Devs可以识别的YAML文件：

```
s cli fc-transform fun2ros --target s.yaml
```

成功执行一键式切换命令后，将在项目目录中生成一个*s.yaml*文件。

在使用一键式切换命令实现迁移时，按需选择以下参数：

- --source：指定使用Funcraft管理的函数资源路径。
- --target：指定迁移后的YAML文件名称。默认迁移后的文件名称是*s.yaml*或*s.yml*，如果您想自定义YAML文件的名称，则需要使用此参数指定。

- --force：如果在迁移的过程中出现一样的文件名称，可以通过该参数强制覆盖文件。
- --region string：指定部署资源的地域。

输出示例：

```
[2021-09-02T10:34:53.909] [INFO ] [FC-TRANSFORM] - Using funcraft yaml: /test/demo/template.yml
[2021-09-02T10:34:53.923] [INFO ] [FC-TRANSFORM] - Reminder serverless devs yaml path: /test/demo/s.yaml
Tips for next step
======================
* Deploy Resources: s deploy -t s.yaml
End of method: fun2ros
```

8.2. 什么是Serverless Devs

Serverless Devs是一个开源开放的Serverless开发者平台，您无需关心底层资源。通过Serverless Devs，您不仅可以可插拔式地使用Serverless的服务和框架，也可以参与组件和插件的开发，提高运维效率。同时，您还可以更简单、快速地开发、创建、测试和部署项目，实现项目全生命周期的管理。

> **说明** 由于后期将不再维护Funcraft，所以如果您的资源是使用Funcraft进行管理的，为了给您带来更好的用户体验，建议您将资源迁移至Serverless Devs进行管理。
> 关于如何将函数计算的相关资源从Funcraft迁移到Serverless Devs进行管理的详细操作，请参见从Funcraft迁移到Serverless Devs。
> 由此带来的不便，敬请谅解！

优势

Serverless Devs与其他开发者工具不同，可以更灵活开放的使用，具有如下优势：

- 全生命周期管理
  拥有创建、开发、调试、部署和运维项目的全生命周期管理能力。
- 安全发布
  通过其他形式对函数进行变更，Serverless Devs可以感知到并安全更新。
- 快速集成
  借助于Serverless Devs的集成性和被集成性，可以与常见的CI/CD平台或工具等集成。
- 可观测性
  拥有完善的可观测性。您可以在客户端通过指标查询和日志查询等命令，从业务的数据指标、执行日志等多重维度实现观测。
- 多模调试
  拥有多模调试方案，可以同时满足开发态、运维态等不同调试需求，例如本地运行、在线运行和端云联调等。

Serverless Devs灵活与开放的使用方法，可以在自动化部署、运维等领域发挥比较大的作用。将Serverless Devs与项目全生命周期进行有机融合，使Serverless项目开发运维效能可提升90%。更多信息，请参见Serverless Devs。

组件

基于Serverless Devs，阿里云函数计算团队为您提供了多种组件，其中，与函数计算相关的组件主要有FC组件和FC-API组件等，说明如下。

FC组件

FC组件是一个用于支持阿里云Serverless应用全生命周期的工具。该组件是Funcraft的进阶版本，和Funcraft的行为描述类似，该工具通过资源配置文件即*s.yaml*，可以快速帮助您开发、构建、测试以及将应用部署到函数计算。

- FC组件的使用形态：
  - FC组件（YAML模式）
    使用阿里云函数计算组件的YAML规范即*s.yaml*文件定义Serverless资源。其包含函数计算服务、函数、触发器以及自定义域名等资源。阿里云函数计算组件的YAML规范的详细信息，请参见规范和权限管理。

- FC组件（非YAML模式）
  FC组件（非YAML模式）即FC组件CLI模式，该模式下您无需在本地创建*s.yaml*文件并配置信息就能实现资源管理。例如同步管理资源，当您已成功将资源部署到线上然后想免除手写配置信息等，您只需执行 `s cli fc sync` 相关命令，就能将云上资源，例如函数配置和代码同步至本地。

- FC组件的优势：
  - 简洁：FC组件使用一键式的部署模式进行创建、更新服务等资源。同时，为了保证组件使用的流畅性，所有的功能均是按需加载。
  - 高效性：FC组件具备开发和运维态能力，支持部署、移除功能，也支持在本地进行构建、端云联调、查询信息、安装依赖和调试等功能，提升开发、测试的效率。
  - 多样化部署：FC组件目前支持Pulumi和SDK部署模式。

> **说明** FC组件适用于所有部署函数计算应用的场景，例如迁移传统框架到函数计算、管理函数等。

关于FC组件的功能介绍和权限管理，请参见Serverless Devs操作命令。

FC-API组件

FC-API组件是一款基于Serverless Devs的阿里云函数计算API操作工具。该组件是fcli的进阶版本，和fcli的行为描述类似，因此称为FC-API组件，所以您可以直接通过交互式命令进行函数计算相关API的使用。

FC-API组件的优势：

- 由于是关于函数计算API的相关操作，所以在使用的过程中更容易管理权限，甚至可以细化到单个接口的权限管理。
- 由于是纯命令式的，不依赖*s.yaml*文件描述资源，所以在自动化流程中，例如CI/CD工具、场景下的使用中更便捷。
- 支持函数管理能力，例如查询服务列表、函数列表或触发器列表，也支持增删改查相关资源。

FC-API组件的适用场景

- 自动化流程
  由于FC-API组件是纯粹通过命令行进行开发的，无需依赖*s.yaml*文件，所以在使用的过程中，可以通过参数的传递实现某些功能，而且这些功能相对来说都是原子性的，更容易做权限管理和自动化流程。您可以直接通过单个命令实现某些操作，例如更新函数代码、更新函数配置、更新服务、创建触发器和更新别名等。
- 管理函数
  FC-API组件是基于阿里云函数计算API的命令行工具，具备已有的资源管理能力。例如可以更新服务、删除服务、查看服务列表、函数列表或触发器列表等。

FC-API组件的基本功能，如下所示：

- 服务相关命令
- 函数相关命令
- 触发器相关命令
- 版本相关命令
- 别名相关命令
- 自定义域名相关命令
- 预留配置相关命令
- 函数异步调用相关命令

关于FC-API组件的权限管理，请参见FC-API组件权限信息。

相关信息

- 安装Serverless Devs
- 配置Serverless Devs

8.3. 安装Serverless Devs

本文介绍如何在macOS、Linux和Windows操作系统内安装Serverless Devs。

安装Serverless Devs

> **说明** 函数计算支持通过命令行工具安装Serverless Devs。

1. 按需选择安装Serverless Devs的方式：
   - 通过包管理器安装
     a. 按需选择以下方式安装Serverless Devs。
        - 通过npm包管理器安装：适用于已经预装npm的Windows、macOS和Linux操作系统。
          执行以下命令安装Serverless Devs。

          ```
          npm install @serverless-devs/s -g
          ```

          > **说明** Node.js需为10.X及以上版本。安装完成后，您可以执行 `node --version` 命令查看Node.js的版本。

        - 通过yarn包管理器安装：适用于已经预装yarn的Windows、macOS和Linux操作系统。
          执行以下命令安装Serverless Devs。

          ```
          yarn global add @serverless-devs/s
          ```

     b. 执行以下命令，验证是否安装成功。

        ```
        s -v
        ```

        成功安装后的执行结果：

        ```
        @serverless-devs/s: 2.0.83, linux-x64, node-v14.17.4
        ```

   - 通过脚本安装：适用于macOS、Linux操作系统。
     a. 执行以下命令安装Serverless Devs。

        ```
        curl -o- -L http://cli.so/install.sh | bash
        ```

     b. 执行以下命令，验证是否安装成功。

        ```
        s -v
        ```

        成功安装后的执行结果：

        ```
        @serverless-devs/s: 2.0.83, linux-x64, node-v14.17.4
        ```

（可选）安装Docker

如果您需要通过Serverless Devs的FC组件进行构建、本地运行调试函数和端云联调等操作，例如执行 `s build` 、 `s local` 和 `s proxied` 等相关命令，则需要在您的开发环境内安装Docker。

根据实际情况选择安装方式：

- Windows操作系统
  请参见官方教程。若遇到网络问题，可以下载阿里云提供的Docker For Windows。
- macOS操作系统
  请参见官方教程。若遇到网络问题，可以下载阿里云提供的Docker For macOS。
- Linux操作系统
  请参见官方教程。若遇到网络问题，请参见阿里云Docker CE镜像源站下载。

常见问题

当您出现以下类似问题时，按需选择以下方式解决：

- 当Serverless Devs的版本比较低时，怎么升级Serverless Devs?
  您可以再次执行安装命令完成升级操作。详细信息，请参见安装Serverless Devs。

- 如果安装过程比较慢，您可以使用淘宝npm源，安装命令如下所示：

```
npm --registry=https://registry.npmmirror.com install @serverless-devs/s -g
```

- 当您在查看Serverless Devs版本时未及时生效，您可以按需选择以下方式解决：
  - 执行以下命令，设置环境变量。

```
export PATH=$HOME/.s/node-v14.19.3-linux-x64/bin:$PATH
```

> **说明** 本命令中 `node-v14.19.3` 中版本号仅为示例，不同操作系统对应的该版本号可能不同，请以实际系统提示为准。

  - 打开一个新的终端。

- 如果在Linux或macOS操作系统内安装Serverless Devs时报错，且报错信息为 `Error: EACCES: permission denied` ，按需选择解决方案：
  - 通过包管理器安装时出现此类报错，请执行以下命令：

```
sudo npm install @serverless-devs/s -g
```

  - 通过脚本安装时出现此类报错，请执行以下命令：

```
sudo curl -o- -L http://cli.so/install.sh | sudo bash
```

- 在Linux或macOS操作系统内执行完安装命令后，如果执行Serverless Devs的相关命令时报错且报错信息为 `Command not found` ，按需选择解决方案：
  - 通过npm包管理器或脚本安装的Serverless Devs在使用相关命令出现此类报错时，需执行以下命令：

```
ln -s $(npm root -g)/@serverless-devs/s/bin/s /usr/local/bin/s
```

  - 通过yarn包管理器安装的Serverless Devs在使用相关命令出现此类报错时，需执行以下命令：

```
ln -s $(yarn global dir)/node_modules/@serverless-devs/s /usr/local/bin/
```

8.4. 配置Serverless Devs

在使用Serverless Devs前，您需要配置Serverless Devs。本文以阿里云密钥为例，介绍如何使用s config相关命令添加、获取和删除密钥。

前提条件

安装Serverless Devs

添加密钥信息

1. 执行以下命令选择云厂商。

```
s config add
```

   执行输出：

```
> Alibaba Cloud (alibaba)
  AWS (aws)
  Azure (azure)
  Baidu Cloud (baidu)
  Google Cloud (google)
  Huawei Cloud (huawei)
  Tencent Cloud (tencent)
(Move up and down to reveal more choices)
```

2. 选择目标云厂商，然后按回车。本文以 `Alibaba Cloud (alibaba)` 为例。

3. 根据交互提示，设置相关密钥信息。

```
Please select a provider: Alibaba Cloud (alibaba)
 Refer to the document for alibaba key:  http://config.devsapp.net/account/alibaba
 AccountID 188077086902****
 AccessKeyID LTAI4G4cwJkK4Rza6xd9****
 AccessKeySecret eCc0GxSpzfq1DVspnqqd6nmYNN****
 Please create alias for key pair. If not, please enter to skip default
    Alias: default
    AccountID: 188077086902****
    AccessKeyID: LTAI4G4cwJkK4Rza6xd9****
    AccessKeySecret: eCc0GxSpzfq1DVspnqqd6nmYNN****
Configuration successful
```

您可以在函数计算控制台**概览**页的**常用信息**区域查看Account ID和AccessKey ID，如下图所示。



Serverless Devs同时也支持命令式密钥添加，例如执行以下命令配置阿里云的密钥信息：

```
s config add --AccessKeyID LTAI4G4cwJkK4Rza6xd9**** --AccessKeySecret  eCc0GxSpzfq1DVspnqqd6nmYNN***
* --AccountID 188077086902**** --access ***
```

> **注意** 为了方便管理多种密钥信息，Serverless Devs支持多密钥管理。您在配置密钥时，可以为每个密钥配置别名信息aliasName，即可以在使用时指定某个密钥实现特定功能，例如：
>
> - 通过密钥进行环境划分，可使用不同的别名绑定不同的账号，然后将资源部署到不同的账号下，例如账号A作为生产，账号B作为预发。
> - 通过对密钥进行权限控制，使不同的密钥拥有不同的权限做不同的工作，例如：
>   - 配置别名为website-access密钥，仅用于部署网站。
>   - 配置别名为fc-access密钥，仅用于部署函数。

获取密钥信息

参数说明：

- -a或--access：指定别名名称。
- -h或--help：显示帮助信息。

执行以下命令，获取密钥信息：

- 获取指定别名的密钥信息。例如-a指定目标别名为default：

```
s config get -a default
```

输出示例：

```
default:
  AccountID: 188077086902****
  AccessKeyID: LTAI4FoAC3JPPWKPZDVg****
  AccessKeySecret: gwL7FuveTjmHC7xyvvwOxP6ysI****
```

- 获取全部的密钥信息：

```
s config get
```

输出示例：

```
default:
  AccountID: 188077086902****
  AccessKeyID: LTAI4G4cwJkK4Rza6xd9****
  AccessKeySecret: eCc0GxSpzfq1DVspnqqd6nmYNN****
default-1:
  AccountID: 188077086902****
  AccessKeyID: LTAI4FoAC3JPPWKPZDVg****
  AccessKeySecret: gwL7FuveTjmHC7xyvvwOxP6ysI****
```

> **说明** 您可以执行 `s config get -h` 命令获取该命令下的更多信息。

删除密钥信息

参数说明：

- -a或--access：指定别名名称。
- -h或--help：显示帮助信息。

执行以下命令，删除密钥，-a指定别名。

```
s config delete -a aliasName
```

8.5. 使用Serverless Devs管理函数资源

Serverless Devs是一个开源开放的Serverless开发者平台，通过该平台您可以快速地开发、创建、测试和部署项目，实现项目的全生命周期管理。本文以部署运行环境是Node.js 12的函数为例，介绍如何使用Serverless Devs下载对应的函数计算应用模板，然后快速开发部署函数应用。

背景信息

本文介绍两种方式初始化项目。当您初次使用Serverless Devs时，建议您使用方式一体验Serverless Devs。如果您想了解关于Serverless Devs的更多信息，例如本地调试、远程调用、删除资源等操作，建议您使用方式二体验。

前提条件

1. 安装Serverless Devs和Docker
2. 配置Serverless Devs

方式一：使用s命令管理资源

1. 执行以下命令，初始化项目。

   ```
   s
   ```

   请根据输出的交互提示，初始化项目。输出示例：

```
 No Serverless-Devs project is currently detected. Do you want to create a new project? Yes//在项目文件内未检测到Serverless-Devs项目。选择是否创建一个项目。
 Serverless Awesome: https://github.com/Serverless-Devs/Serverless-Devs/blob/master/docs/zh/awesome.md
 Hello Serverless for Cloud Vendors Alibaba Cloud Serverless//选择目标云厂商，本示例以阿里云为例。
 Hello, serverlesser. Which template do you like? Quick start [Deploy a Hello World function to FaaS]//选择您需要创建的模板。本示例以快速部署运行环境为Node.js 12的事件函数为例。
 Which template do you like? [Event] Node.js 12
 Create application command: [s init devsapp/start-fc-event-nodejs12]
 Please input your project name (init dir) start-fc-event-nodejs12//Serverless Devs默认为您生成一个项目名称，您也可以自定义该名称。
 file decompression completed

     ____  _      _ ___  _ _      _         _____ ____
    /  _ \/ \    / \\  \/// \ /\/ \  /|  /    //   _\
    | / \|| |    | | \  / | | |||  |\ ||  |  __\|  /
    | |-||| |_/\| | / /  | \_/|| | \||  | |    |  \__
    \_/ \|\____/\_//_/   \____/\_/  \|  \_/    \____/
 please select credential alias default//选择您在配置Serverless Devs时设置的别名。
    Welcome to the Aliyun FC start application
     This application requires to open these services:
         FC : https://fc.console.aliyun.com/
     * 项目初始化完成，您可以直接进入项目目录下，并使用s deploy进行项目部署。
 Thanks for using Serverless-Devs
 You could [cd /test/start-fc-event-nodejs12] and enjoy your serverless journey!
 If you need help for this example, you can use [s -h] after you enter folder.
 Document Star: https://github.com/Serverless-Devs/Serverless-Devs
 Do you want to deploy the project immediately? Yes//选择是否部署项目，本示例以选择部署项目为例。
 ......
helloworld:
  region:   cn-hangzhou
  service:
    name: hello-world-service
  function:
    name:       event-nodejs12
    runtime:    nodejs12
    handler:    index.handler
    memorySize: 128
    timeout:    60
```

方式二：使用s init命令管理资源

1. 初始化项目。

   在目标文件内执行以下命令，初始化项目：

   ```
   s init devsapp/start-fc-http-nodejs12
   ```

   > **说明** 本示例以部署一个运行环境为Node.js 12的HTTP函数为例，介绍如何快速初始化一个项目。当需要部署其他运行环境的示例时，您可以将 `http` 替换为其他类型的触发方式例如 `event` ，也可以将 `nodejs12` 替换为其他类型的运行环境。

   执行输出：

```
 Serverless Awesome: https://github.com/Serverless-Devs/Serverless-Devs/blob/master/docs/zh/awesome.
md
 Please input your project name (init dir) start-fc-http-nodejs12//Serverless Devs会默认生成一个项目名
称，您也可以自定义该名称。
 file decompression completed

    ____  _     _ ___  _ _     _        _____ ____
   /  _ \/ \   / \\  \/// \ /\/ \  /|  /    //   _\
   | / \|| |   | | \  / | | ||| |\ ||  |  __\|  /
   | |-||| |_/\| | / /  | \_/|| | \||  | |   |  \__
   \_/ \|\____/\_//_/   \____/\_/  \|  \_/   \____/
 please select credential alias default//选择您在配置Serverless Devs时设置的别名。
    Welcome to the Aliyun FC start application
     This application requires to open these services:
         FC : https://fc.console.aliyun.com/
     * 项目初始化完成，您可以直接进入项目目录下，并使用s deploy进行项目部署。
 Thanks for using Serverless-Devs
 You could [cd /test/start-fc-http-nodejs12] and enjoy your serverless journey!
 If you need help for this example, you can use [s -h] after you enter folder.
 Document Star: https://github.com/Serverless-Devs/Serverless-Devs
 Do you want to deploy the project immediately? No//选择部署项目。
```

当成功初始化项目后，会在当前目录中生成一个*start-fc-http-nodejs12*文件，该文件内包含以下文件：

- *s.yaml*：函数计算资源，该资源是以YAML规范定义的。关于YAML规范的详细信息，请参见YAML规范。
- *code*：*code*文件夹内包含*index.js*文件，可在该文件内编辑函数代码逻辑。
- *readme.md*：依赖清单文件，该文件内定义了项目所需的各个模块和项目的配置信息。

2. 执行以下命令，进入项目目录内：

```
cd start-fc-http-nodejs12
```

3. （可选）构建应用。

在项目目录内执行以下命令，构建应用：

```
s build
```

执行成功该步骤后，会在当前目录中生成一个*.s*文件，存放输出的构建产物，该文件可用于本地调试和部署。

4. （可选）本地调试应用。

当您在本地完成应用的开发构建后，可能需要在本地调试验证功能是否正常。您可以按需选择命令调试函数：

- HTTP函数

```
s local start
```

执行输出：

```
[2021-12-16 07:17:05] [INFO] [FC-LOCAL-INVOKE] - Using trigger for start: name: httpTrigger
type: http
config:
  authType: anonymous
  methods:
    - GET
[2021-12-16 07:17:05] [INFO] [FC-LOCAL-INVOKE] - HttpTrigger httpTrigger of hello-world-service/h
ttp-trigger-nodejs12 was registered
        url: http://localhost:7013/2016-08-15/proxy/hello-world-service/http-trigger-nodejs12/
        methods: GET
        authType: anonymous
Tips: you can also use these commands to run/debug custom domain resources:
Start with customDomain:
* s local start auto
Debug with customDomain:
* s local start -d 3000 auto
Tips for next step
======================
* Deploy Resources: s deploy
helloworld:
  status: succeed
function compute app listening on port 7013!
```

- 事件函数

```
s local invoke
```

当您执行调试命令后，由于函数类型不同返回结果也会不同：

- HTTP函数
  当您的函数是HTTP函数时，执行完调试命令后，将在执行输出中返回一个URL地址，此时调试流程阻塞在这里，您可以使用以下方式触发函数执行：
  - 使用cURL工具触发函数执行

    ```
    curl http://localhost:7013/2016-08-15/proxy/hello-world-service/http-trigger-nodejs12/
    ```

  - 使用浏览器访问HTTP函数

    ```
    http://localhost:7013/2016-08-15/proxy/hello-world-service/http-trigger-nodejs12/
    ```

- 事件函数
  如果您的函数是事件函数，执行完调试命令后，将立即返回调试结果。

5. 部署应用。

   执行以下命令，将应用部署至函数计算：

   ```
   s deploy -y
   ```

   执行输出：

```
 Checking Service, Function, Triggers (0.82s)
 Creating Service, Function, Triggers (0.6s)
 Creating custom domain (0.31s)
Tips for next step
======================
* Display information of the deployed resource: s info
* Display metrics: s metrics
* Display logs: s logs
* Invoke remote function: s invoke
* Remove Service: s remove service
* Remove Function: s remove function
* Remove Trigger: s remove trigger
* Remove CustomDomain: s remove domain
helloworld:
  region:   cn-hangzhou
  service:
    name: hello-world-service
  function:
    name:       http-trigger-nodejs12
    runtime:    nodejs12
    handler:    index.handler
    memorySize: 128
    timeout:    60
  url:
    custom_domain:
      -
        domain: http://http-trigger-nodejs12.hello-world-service.188077086902****.cn-hangzhou.fc.dev
sapp.net
  triggers:
    -
      type: http
      name: httpTrigger
```

执行成功后，该命令会将应用部署至函数计算。

6. （可选）远程调试应用。

   执行以下命令，远程调试应用：

```
s invoke -e '{"body": "","method":"GET","headers":{"header_1":"v1"},"queries":{"key":"value"},"path"
:"/"}'
```

   执行输出：

```
Request url: https://188077086902****.cn-hangzhou.fc.aliyuncs.com/2016-08-15/proxy/hello-world-servi
ce/http-trigger-nodejs12/
========= FC invoke Logs begin =========
FC Invoke Start RequestId: 6a3adedb-5d5d-4ac5-aa0b-2d43f4ace65c
load code for handler:index.handler
2022-03-04T08:39:14.770Z 6a3adedb-5d5d-4ac5-aa0b-2d43f4ace65c [verbose] hello world
FC Invoke End RequestId: 6a3adedb-5d5d-4ac5-aa0b-2d43f4ace65c
Duration: 66.23 ms, Billed Duration: 67 ms, Memory Size: 128 MB, Max Memory Used: 42.48 MB
========= FC invoke Logs end =========
FC Invoke Result[Code: 200]:
{
  path: '//',
  queries: { key: 'value' },
  headers: {
    accept: 'application/json',
    authorization: 'FC LTAI4G4cwJkK4Rza6xd9****:KZsoanKle4SoRO8EjyOjY3duLlVlR1mp78hjV9WG****',
    date: 'Fri, 04 Mar 2022 08:39:14 GMT',
    header_1: 'v1',
    host: '188077086902****.cn-hangzhou.fc.aliyuncs.com',
    'user-agent': 'Node.js(v14.17.4) OS(linux/x64) SDK(@alicloud/fc2@v2.5.0)',
    'x-forwarded-proto': 'http'
  },
  method: 'GET',
  requestURI: '/2016-08-15/proxy/hello-world-service/http-trigger-nodejs12//?key=value',
  clientIP: '47.97.XX.XX',
  body: ''
}
End of method: invoke
```

由于本示例部署的函数是HTTP函数，所以在成功执行远程调试命令后，执行输出会返回一个URL，您可以使用cURL工具触发执行线上的函数：

```
curl https://188077086902****.cn-hangzhou.fc.aliyuncs.com/2016-08-15/proxy/hello-world-service/http-
trigger-nodejs12/
```

7. （可选）删除应用。

   执行以下命令删除应用：

   ```
   s remove service
   ```

   执行输出：

   ```
   Need to delete the resource in the cn-hangzhou area, the operation service is hello-world-service:
    ......
    Delete trigger hello-world-service/http-trigger-nodejs12/httpTrigger success.
    Delete function hello-world-service/http-trigger-nodejs12 success.
    Delete service hello-world-service success.
   [2022-03-04 08:40:26] [INFO] [FC] - Getting list on-demand: _FC_NAS_hello-world-service-ensure-nas-d
   ir-exist-service
   [2022-03-04 08:40:26] [INFO] [FC] - Getting list provision: _FC_NAS_hello-world-service-ensure-nas-d
   ir-exist-service
   [2022-03-04 08:40:26] [INFO] [FC] - Getting listAliases: _FC_NAS_hello-world-service-ensure-nas-dir-
   exist-service
   [2022-03-04 08:40:26] [INFO] [FC] - Getting list on-demand: _FC_NAS_hello-world-service
   [2022-03-04 08:40:26] [INFO] [FC] - Getting list provision: _FC_NAS_hello-world-service
   [2022-03-04 08:40:26] [INFO] [FC] - Getting listAliases: _FC_NAS_hello-world-service
   End of method: remove
   ```

   当成功执行该命令时，表示应用已被删除。

8.6. Serverless Devs操作命令

Serverless Devs是一个开源开放的Serverless开发者平台。通过Serverless Devs，您可以实现项目全生命周期的管理。本文介绍Serverless Devs在实现全生命周期的过程中涉及的规范、权限管理和命令列表。

规范和权限管理

您也可以通过以下信息查看YAML规范中相关字段的详细信息和权限策略。



YAML规范

您可以通过YAML规范设置s.yaml文件的参数信息配置函数资源。



YAML规范

您可以通过YAML规范设置s.yaml文件的参数信息配置函数资源。



service



function



triggers



customDomains



service



function



triggers



命令列表

构建和部署	可观测性	调用和调试	发布和配置	其他
构建函数或安装依赖（s build）	查询指标信息（s metrics）	本地调用（s local）	函数版本操作（s version）	将资源从Funcraft迁移至Serverless Devs管理
部署资源（s deploy）	查询日志信息（s logs）	线上调试或触发（s invoke）	函数别名操作（s alias）	挂载硬盘（s nas）
删除资源（s remove）	无	端云联调（s proxied）	函数预留操作（s provision）	查询资源（s info）
获取变更资源信息（s plan）	无	登录实例（s instance）	按量资源（s onDemand）	同步资源（s sync）
无	无	Power Tuning（s eval）	层（s layer）	压测（s stress）

更多信息

集成CI/CD平台或工具
Serverless Devs拥有集成性和被集成性，例如可以通过Github Action、GiteeGo、Jenkins或云效和Serverless Devs实现CI/CD能力。

集成CI/CD平台或工具
Serverless Devs拥有集成性和被集成性，例如可以通过Github Action、GiteeGo、Jenkins或云效和Serverless Devs实现CI/CD能力。

8.7. FC-API组件

8.7.1. FC-API组件权限信息

本文列举使用FC-API组件涉及的自定义权限策略。

Resource	Action	描述
acs:fc:<region>:<account-id>:services/<serviceName>	fc:GetService	特定服务资源。
	fc:UpdateService	
	fc:DeleteService	
acs:fc:<region>:<account-id>:services/*	fc:CreateService	所有服务资源。
	fc:ListServices	

Resource	Action	描述
acs:fc:<region>:<account-id>:services/<serviceName>.<qualifier>	fc:GetService	特定版本的服务资源。
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>	fc:GetFunction	特定服务下的特定函数资源。
	fc:UpdateFunction	
	fc:DeleteFunction	
	fc:InvokeFunction	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/*	fc:CreateFunction	特定服务下的所有函数资源。
	fc:ListFunctions	
acs:fc:<region>:<account-id>:services/<serviceName>.*/functions/<functionName>	fc:GetFunction	特定服务的所有版本下的所有函数资源。
	fc:UpdateFunction	
	fc:DeleteFunction	
	fc:InvokeFunction	
	fc:PutProvisionConfig	
	fc:GetProvisionConfig	
	fc:PutFunctionOnDemandConfig	
	fc:DeleteFunctionOnDemandConfig	
	fc:PutFunctionAsyncInvokeConfig	
	fc:DeleteFunctionAsyncInvokeConfig	
	fc:GetFunctionAsyncInvokeConfig	
	fc:GetFunctionOnDemandConfig	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>/triggers/<triggerName>	fc:GetTrigger	特定服务下的特定函数下的特定触发器资源。
	fc:UpdateTrigger	
	fc:DeleteTrigger	
acs:fc:<region>:<account-id>:services/<serviceName>/functions/<functionName>/triggers/*	fc:CreateTrigger	特定服务下的特定函数下的所有触发器资源。
	fc:ListTriggers	
acs:fc:<region>:<account-id>:services/<serviceName>/versions	fc:PublishServiceVersion	所有版本。
	fc:ListServiceVersions	
acs:fc:<region>:<account-id>:services/<serviceName>/versions/<versionID>	fc:DeleteServiceVersion	指定版本。

Resource	Action	描述
acs:fc:<region>:<account-id>:services/<serviceName>/aliases/*	fc:CreateAlias	所有别名。
	fc:ListAliases	
acs:fc:<region>:<account-id>:services/<serviceName>/aliases/<aliasName>	fc:GetAlias	指定别名。
	fc:UpdateAlias	
	fc:DeleteAlias	
acs:fc:<region>:<account-id>:custom-domains/*	fc:CreateCustomDomain	所有自定义域名。
	fc:ListCustomDomains	
acs:fc:<region>:<account-id>:custom-domains/<domainName>	fc:GetCustomDomain	指定自定义域名。
	fc:UpdateCustomDomain	
	fc:DeleteCustomDomain	
acs:fc:<region>:<account-id>:tag	fc:TagResource	单个标签。
	fc:GetResourceTags	
	fc:UnTagResource	
acs:fc:<region>:<account-id>:tags/*	fc:ListTaggedResources	所有标签。
acs:fc:<region>:<account-id>:account-settings/*	fc:GetAccountSettings	用户设置。
acs:fc:<region>:<account-id>:layerarn/:arn	fc:GetLayerVersionByArn	所有层。
acs:fc:<region>:<account-id>:layers/*	fc:ListLayers	
acs:fc:<region>:<account-id>:layers/:layerName/versions/:version	fc:PublishLayerAsPublic	
acs:fc:<region>:<account-id>:layers/:layerName/versions/*	fc:ListLayerVersions	所有层版本。
	fc:CreateLayerVersion	
acs:fc:<region>:<account-id>:layers/:layerName/versions/:version	fc:GetLayerVersion	
	fc:DeleteLayerVersion	
acs:fc:<region>:<account-id>:on-demand-configs/*	fc:ListOnDemandConfigs	按量实例配置。
acs:fc:<region>:<account-id>:provision-configs/*	fc:ListProvisionConfigs	预留实例配置。
acs:fc:<region>:<account-id>:services/:serviceName/binding	fc:DeleteVpcBinding	VPC绑定

Resource	Action	VPC绑定。 描述
acs:fc:<region>:<account-id>:services/:serviceName/binding/*	fc:CreateVpcBinding	
	fc:ListVpcBindings	
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/async-invoke-configs/*	fc:ListFunctionAsyncInvokeConfigs	异步调用配置。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/code	fc:GetFunctionCode	所有函数代码。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/stateful-async-invocations/*	fc:ListStatefulAsyncInvocations	有状态异步调用。
acs:fc:<region>:<account-id>:services/:serviceName/functions/:functionName/stateful-async-invocations/:invocationId	fc:GetStatefulAsyncInvocation	
	fc:StopStatefulAsyncInvocation	

更多信息
完整的权限信息，请参见权限策略及示例。

8.7.2. 服务相关命令
本文介绍FC-API组件中服务的相关命令。

前提条件
您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs

创建服务
命令格式：

```
s cli fc-api createService --region <regionid> --access <accessname> --props <props> --serviceName <serviceName> --description <description> --internetAccess <true or false> --role <ram role> --logConfig <logconfig> --nasConfig <nasconfig> --vpcConfig <vpcconfig> --tracingConfig <tracingconfig>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --description string：指定服务中的描述信息。
- --internetAccess string：配置是否允许函数访问互联网。取值为 `true` 或 `false` 。
- --role string：指定RAM角色。
- --logConfig string：日志配置。配置函数产生的日志可以写入的Logstore。
- --nasConfig string：NAS配置。指定函数可以访问的NAS。
- --vpcConfig string：VPC配置。指定函数可以访问的VPC。

- --tracingConfig string：配置是否开启链路追踪。取值为 `Enable` 或 `Disable` 。

执行示例：

```
s cli fc-api createService --region cn-hangzhou --serviceName mytest --description my description --logC
onfig my-log-store --tracingConfig Enable
```

关于创建服务的API接口的详细信息，请参见CreateService。

删除服务

命令格式：

```
s cli fc-api deleteService --region <regionid> --access <accessname> --props <props> --serviceName <serv
iceName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。

执行示例：

```
s cli fc-api deleteService --region cn-hangzhou --access default --serviceName mytest
```

关于删除服务的API接口的详细信息，请参见DeleteService。

更新服务

命令格式：

```
s cli fc-api updateService --region <regionid> --access <accessname> --props <props> --serviceName <serv
iceName> --description <description> --internetAccess <true or false> --role <ram role> --logConfig <log
config> --nasConfig <nasconfig> --vpcConfig <vpcconfig> --tracingConfig <tracingconfig>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --description string：指定服务中的描述信息。
- --internetAccess string：配置是否允许函数访问互联网。取值为 `true` 或 `false` 。
- --role string：指定RAM角色。
- --logConfig string：日志配置。配置函数产生的日志可以写入的Logstore。
- --nasConfig string：NAS配置。指定函数可以访问的NAS。
- --vpcConfig string：VPC配置。指定函数可以访问的VPC。
- --tracingConfig string：配置是否开启链路追踪。取值为 `Enable` 或 `Disable` 。

执行示例：

```
s cli fc-api updateService --region cn-hangzhou --serviceName mytest --description my description --logC
onfig my-log-store --tracingConfig Disable
```

关于更新服务的API接口的详细信息，请参见UpdateService。

获取服务配置信息

命令格式：

```
s cli fc-api getService --region <regionid> --access <accessname> --props <props> --serviceName <service
Name> --qualifier <version or alias>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api getService --region cn-hangzhou --access default --serviceName mytest --qualifier LATEST
```

关于获取服务配置信息的API接口的详细信息，请参见GetService。

查询服务列表

命令格式：

```
s cli fc-api listServices --region <regionid> --access <accessname>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。

执行示例：

```
s cli fc-api listServices --region cn-hangzhou --access default
```

关于获取服务列表的API接口的详细信息，请参见ListServices。

8.7.3. 函数相关命令

本文介绍FC-API组件中函数的相关命令。

前提条件

您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务

创建函数

命令格式：

```
s cli fc-api createFunction --region <regionid> --access <accessname> --props <props> --serviceName <ser
viceName> --functionName <functionName> --description <description> --code <codefile> --customContainerC
onfig <customContainerConfig>  --handler <functionhandler> --runtime <runtime> --initializationTimeout <
initializationTimeout> --initializer ----memorySize <momorysize> ----timeout <timeout> --caPort <caport>
```

参数说明：

- --region string：指定部署资源的地域。

- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --code string：指定压缩的代码文件，代码文件必须是ZIP格式。
- --customContainerConfig string：配置Custom Container运行时的参数。配置后函数可以使用自定义容器镜像执行函数。
- --description string：指定函数中的描述信息。
- --handler string：设置函数handler，handler的格式为“文件名.函数名”。例如 `hello_world.handler` 指定了函数的调用入口为*hello_world.js*文件中的handler函数。
- --initializationTimeout number：设置初始化函数超时时间。
- --initializer string：设置初始化函数。
- --memorySize number：设置函数执行的内存大小。
- --runtime string：指定运行环境。当前支持 `nodejs6` 、 `nodejs8` 、 `nodejs10` 、 `nodejs12` 、 `python2.7` 、 `python3` 、 `java8` 、 `java11` 、 `php7.2` 、 `dotnetcore2.1` 、 `custom runtime` 及 `custom container` 。
- --timeout number：设置函数执行的超时时间（默认60s）。
- --caPort number：指定Custom Runtime或Custom Container运行时HTTP Server的监听端口。

执行示例：

- macOS操作系统或Linux操作系统，根据运行环境的不同选择执行示例：
  - 当您的运行环境是编程语言运行时，请执行以下命令创建函数：

    ```
    s cli fc-api createFunction --serviceName mytest --functionName mytest --code '{"zipFile": "./"}' --handler index.handler --runtime python3
    ```

  - 当您的运行环境是自定义容器时，请执行以下命令创建函数：

    ```
    s cli fc-api createFunction --serviceName mytest --functionName mytest --customContainerConfig '{"image": "registry-vpc.cn-shenzhen.aliyuncs.com/fc-demo/flask:v0.1"}' --handler index.handler --runtime custom-container
    ```

- Windows操作系统
  - 当您的运行环境是编程语言运行时，请执行以下命令创建函数：

    ```
    s cli fc-api createFunction --serviceName mytest --functionName mytest --code "{\"zipFile\": \"./\"}" --handler index.handler --runtime python3
    ```

  - 当您的运行环境是自定义容器时，请执行以下命令创建函数：

    ```
    s cli fc-api createFunction --serviceName mytest --functionName mytest --customContainerConfig "{\"image\": \"registry-vpc.cn-shenzhen.aliyuncs.com/fc-demo/flask:v0.1\"}" --handler index.handler --runtime custom-container
    ```

关于创建函数的API接口的详细信息，请参见CreateFunction。

删除函数

命令格式：

```
s cli fc-api deleteFunction --region <regionid> --access <accessname> --props <props> --serviceName <serviceName> --functionName <functionName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。

- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。

执行示例：

```
s cli fc-api deleteFunction --region cn-hangzhou --access default --serviceName mytest --functionName my
test
```

关于删除函数的API接口的详细信息，请参见DeleteFunction。

更新函数

命令格式：

```
s cli fc-api updateFunction --region <regionid> --access <accessname> --props <props> --serviceName <ser
viceName> --functionName <functionName> --description <description> --code <codefile> --customContainerC
onfig <customContainerConfig>  --handler <functionhandler> --runtime <runtime> --initializationTimeout <
initializationTimeout> --initializer ----memorySize <momorysize> ----timeout <timeout> --caPort <caport>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --code string：指定压缩的代码文件，代码文件必须是ZIP格式。
- --customContainerConfig string：配置Custom Container运行时的参数。配置后函数可以使用自定义容器镜像执行函数。
- --description string：指定函数中的描述信息。
- --handler string：设置函数handler，handler的格式为“文件名.函数名”。例如 `hello_world.handler` 指定了函数的调用入口为*hello_world.js*文件中的handler函数。
- --initializationTimeout number：设置初始化函数超时时间。
- --initializer string：设置初始化函数。
- --memorySize number：设置函数执行的内存大小。
- --runtime string：指定运行环境。当前支持 `nodejs6` 、 `nodejs8` 、 `nodejs10` 、 `nodejs12` 、 `python2.7` 、 `python3` 、 `java8` 、 `java11` 、 `php7.2` 、 `dotnetcore2.1` 、 `custom runtime` 及 `custom container` 。
- --timeout number：设置函数执行的超时时间（默认60s）。
- --caPort number：指定Custom Runtime或Custom Container运行时HTTP Server的监听端口。

执行示例：

- macOS操作系统或Linux操作系统
  - 当您的运行环境是编程语言运行时，请执行以下命令更新函数：

    ```
    s cli fc-api updateFunction --serviceName mytest --functionName mytest --code '{"zipFile": "./"}' -
    -timeout 31
    ```

  - 当您的运行环境是自定义容器时，请执行以下命令更新函数：

    ```
    s cli fc-api updateFunction --serviceName mytest --functionName mytest --code '{"zipFile": "./"}' -
    -timeout 31 --customContainerConfig '{"image": "registry-vpc.cn-shenzhen.aliyuncs.com/fc-demo/flask
    :v0.1"}'
    ```

- Windows操作系统

- 当您的运行环境是编程语言运行时，请执行以下命令更新函数：

```
s cli fc-api updateFunction --serviceName mytest --functionName mytest --code "{\"zipFile\": \"./\"
}" --timeout 31
```

- 当您的运行环境是自定义容器时，请执行以下命令更新函数：

```
s cli fc-api updateFunction --serviceName mytest --functionName mytest --code "{\"zipFile\": \"./\"
}" --timeout 31 --customContainerConfig "{\"image\": \"registry-vpc.cn-shenzhen.aliyuncs.com/fc-dem
o/flask:v0.1\"}"
```

关于更新函数的API接口的详细信息，请参见UpdateFunction。

获取函数配置信息

命令格式：

```
s cli fc-api getFunction --region <regionid> --access <accessname> --props <props> --serviceName <servic
eName> --functionName <functionName> --qualifier <version or alias>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api getFunction --serviceName mytest --functionName mytest --qualifier LATEST
```

关于获取函数配置信息的API接口的详细信息，请参见GetFunction。

获取函数代码信息

命令格式：

```
s cli fc-api getFunctionCode --region <regionid> --access <accessname> --props <props> --serviceName <se
rviceName> --functionName <functionName> --qualifier <version or alias>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api getFunctionCode --region cn-hangzhou --serviceName mytest --functionName mytest --qualifier
LATEST
```

关于获取函数代码信息的API接口的详细信息，请参见GetFunctionCode。

查询函数列表

命令格式：

```
s cli fc-api listFunctions --region <regionid> --access <accessname> --props <props> --serviceName mytes
t --qualifier <version or alias> --limit <limit number> --nextToken <nextresult> --prefix <return prefix
> --startkey <startkey>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。
- --prefix string：设置返回资源的名称前缀。
- --startKey string：设定结果从startKey之后（包括startKey）按字母排序的第一个开始返回。
- --serviceName string：指定服务名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api listFunctions --serviceName mytest --qualifier LATEST
```

关于获取函数列表的API接口的详细信息，请参见ListFunctions。

调用执行函数

命令格式：

```
s cli fc-api invokeFunction --region <regionid> --access <accessname> --props <props> --serviceName <ser
viceName> --functionName <functionName> --event <event>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --event string：设置触发函数执行的事件。

执行示例：

```
s cli fc-api invokeFunction --region cn-hangzhou --serviceName mytest --functionName mytest
```

关于调用执行函数的API接口的详细信息，请参见InvokeFunction。

8.7.4. 触发器相关命令

本文介绍FC-API组件中触发器相关的命令。

前提条件

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务
- 创建函数

- （可选）创建存储空间
  创建OSS触发器时需要创建存储空间。

  > **说明** OSS Bucket和您的函数需要在同一地域。

- （可选）创建主题
  创建MNS触发器时需要创建主题。
- （可选）创建Tablestore触发器时需要完成以下操作：
  i. 创建Tablestore实例
  ii. 创建数据表
- （可选）创建CDN触发器时，您需要完成以下操作：
  - 创建存储空间
  - 添加加速域名

创建触发器

命令格式：

```
s cli fc-api createTrigger --region <cn-hangzhou> --access <keyname> --props <props> --serviceName <serv
iceName> --functionName <functionName> --invocationRole <role> --qualifier <version or alias> --triggerN
ame <triggerName>  --triggerConfig <triggerconfig> --triggerType <triggertype>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --invocationRole string：触发器调用函数时所需的角色。
- --qualifier string：指定服务的别名或版本。
- --sourceArn string：触发器事件源的ARN。
- --triggerConfig string：触发器配置。
- --triggerName string：指定触发器名称。
- --triggerType string：指定触发器类型。触发器类型支持OSS触发器、日志服务触发器、定时触发器、HTTP触发器、Tablestore触发器、CDN事件触发器和MNS主题触发器。

执行示例：

- macOS操作系统或Linux操作系统

```
s cli fc-api createTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerN
ame mytrigger  --triggerConfig '{"payload": "", "cronExpression": "@every 1m", "enable": true}' --tri
ggerType timer
```

- Windows操作系统

```
s cli fc-api createTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerN
ame mytrigger  --triggerConfig "{\"payload\": \"\", \"cronExpression\": \"@every 1m\", \"enable\": tr
ue}" --triggerType timer
```

关于创建触发器的API接口的详细信息，请参见CreateTrigger。

删除触发器

命令格式：

```
s cli fc-api deleteTrigger --region <regionid> --aeecss <accessname> --props <props> --serviceName <serv
iceName> --functionName <functionName> --triggerName <triggerName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --triggerName string：指定触发器名称。

执行示例：

```
s cli fc-api deleteTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerName
mytrigger
```

关于删除触发器的API接口的详细信息，请参见DeleteTrigger。

更新触发器

命令格式：

```
s cli fc-api updateTrigger --region <regionid> --serviceName <serviceName> --functionName <functionName>
--triggerName <triggerName>  --triggerConfig <triggerconfig> --triggerType <triggertype>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --invocationRole string：触发器调用函数时所需的角色。
- --qualifier string：指定服务的别名或版本。
- --sourceArn string：触发器事件源的ARN。
- --triggerConfig string：触发器配置。
- --triggerName string：指定触发器名称。
- --triggerType string：指定触发器类型。触发器类型支持OSS触发器、日志服务触发器、定时触发器、HTTP触发器、Tablestore触发器、CDN事件触发器和MNS主题触发器。

执行示例：

- macOS操作系统或Linux操作系统

```
s cli fc-api updateTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerN
ame mytrigger  --triggerConfig '{"payload": "", "cronExpression": "@every 1m", "enable": true}' --tri
ggerType timer
```

- Windows操作系统

```
s cli fc-api updateTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerN
ame mytrigger  --triggerConfig "{\"payload\": \"\", \"cronExpression\": \"@every 1m\", \"enable\": tr
ue}" --triggerType timer
```

关于更新触发器的API接口的详细信息，请参见UpdateTrigger。

获取触发器配置信息

命令格式：

```
s cli fc-api getTrigger --region <regionid> --access <accessName> --props <props> --serviceName <service
Name> --functionName <functionName> --triggerName <triggerName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --triggerName string：指定触发器名称。

执行示例：

```
s cli fc-api getTrigger --region cn-hangzhou --serviceName mytest --functionName mytest --triggerName my
trigger
```

关于获取触发器配置信息的API接口的详细信息，请参见GetTrigger。

查询触发器列表

命令格式：

```
s cli fc-api listTriggers --region <regionid> --access <accessname> --props <props> --serviceName <servi
ceName> --functionName <functionName> --limit <limit number> --nextToken <nextresult> --prefix <return p
refix> --startkey <startkey>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。
- --prefix string：设置返回资源的名称前缀。
- --startKey string：设定结果从startKey之后（包括startKey）按字母排序的第一个开始返回。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。

执行示例：

```
s cli fc-api listTriggers --serviceName mytest --functionName mytest
```

关于查询触发器列表的API接口的详细信息，请参见ListTriggers。

8.7.5. 版本相关命令

本文介绍FC-API组件中版本相关的命令。

前提条件

您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs

- 创建服务

创建版本

命令格式：

```
s cli fc-api publishVersion --region <regionid> --access <accessname> --props <props> --serviceName <ser
viceName> --description <description>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --description string：指定服务中的描述信息。

执行示例：

```
s cli fc-api publishVersion --serviceName mytest --description 'mytest'
```

关于创建版本的API接口的详细信息，请参见PublishServiceVersion。

删除版本

命令格式：

```
s cli fc-api deleteVersion --region <regionid> --access <accessname> --props <props> --serviceName <serv
iceName> --versionId <version>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --versionId string：指定服务的版本。

执行示例：

```
s cli fc-api deleteVersion --region cn-hangzhou --serviceName mytest --versionId 1
```

关于删除版本的API接口的详细信息，请参见DeleteServiceVersion。

查询版本

命令格式：

```
s cli fc-api listVersions --region <regionid> --props <props> --access <accessname>  --limit <limit numb
er> --serviceName <serviceName> --prefix <return prefix> --nextToken <nextresult> --startkey <startkey>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。

- --prefix string：设置返回资源的名称前缀。
- --startKey string：设定结果从startKey之后（包括startKey）按字母排序的第一个开始返回。
- --serviceName string：指定服务名称。

执行示例：

```
s cli fc-api listVersions --access default --serviceName mytest --prefix test
```

关于查询版本的API接口的详细信息，请参见ListServiceVersions。

8.7.6. 别名相关命令

本文介绍FC-API组价中别名相关的命令。

前提条件

您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务
- 创建版本

创建别名

命令格式：

```
s cli fc-api createAlias --region <regionid> --access <accessname> --props <props> --serviceName <servic
eName> --versionId <versionid> --aliasName <aliasname> --description <description> --additionalVersionWe
ight <additionalVersionWeight>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --aliasName string：指定别名名称。
- --versionId string：指定别名指向的版本。
- --additionalVersionWeight string：设置别名指向的灰度版本和灰度权重。
- --description string：指定别名的描述信息。

执行示例：

- macOS操作系统或Linux操作系统

  ```
  s cli fc-api createAlias --serviceName mytest --versionId 1 --aliasName release --additionalVersionWe
  ight '{"2": 0.2}'
  ```

- Windows操作系统

  ```
  s cli fc-api createAlias --serviceName mytest --versionId 1 --aliasName release --additionalVersionWe
  ight "{\"1\": 0.2}"
  ```

关于创建别名的API接口的详细信息，请参见CreateAlias。

删除别名

命令格式：

```
s cli fc-api deleteAlias --region <regionid> --access <accessname> --props <props> --serviceName <servic
eName> --aliasName <ailasName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --aliasName string：指定别名名称。

执行示例：

```
s cli fc-api deleteAlias --region cn-hangzhou --serviceName mytest --aliasName release
```

关于删除别名的API接口的详细信息，请参见DeleteAlias。

更新别名

命令格式：

```
s cli fc-api updateAlias --region <regionid> --access <accessname> --props <props> --serviceName <servic
eName> --aliasName <aliasName> --description <description> --versionId <versionid> --additionalVersionWe
ight <additionalVersionWeight>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --aliasName string：指定别名名称。
- --versionId string：指定别名指向的版本。
- --additionalVersionWeight string：设置别名指向的灰度版本和灰度权重。
- --description string：指定别名的描述信息。

执行示例：

- macOS操作系统或Linux操作系统

  ```
  s cli fc-api updateAlias --aliasName release --description tttt --versionId '1' --additionalVersionWe
  ight '{"2": 0.1}' --serviceName mytest
  ```

- Windows操作系统

  ```
  s cli fc-api updateAlias --aliasName release --description tttt --versionId "1" --additionalVersionWe
  ight "{\"1\": 0.1}" --serviceName mytest
  ```

关于更新别名的API接口的详细信息，请参见UpdateAlias。

获取别名配置信息

命令格式：

```
s cli fc-api getAlias --region <regionid> --access <accessname> --props <props> --aliasName <aliasName>
--serviceName <serviceName>
```

参数说明：

- --region string：指定部署资源的地域。

- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --aliasName string：指定别名名称。

执行示例：

```
s cli fc-api getAlias --aliasName release --serviceName mytest
```

关于获取别名配置信息的API接口的详细信息，请参见GetAlias。

获取别名列表

命令格式：

```
s cli fc-api listAliases --region <regionid> --access <accessname> --props <props> --serviceName <servic
eName> --limit <limit number> --nextToken <nextresult> --prefix <return prefix> --startkey <startkey>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。
- --prefix string：设置返回资源的名称前缀。
- --startKey string：设定结果从startKey之后（包括startKey）按字母排序的第一个开始返回。
- --serviceName string：指定服务名称。

执行示例：

```
s cli fc-api listAliases --region cn-hangzhou --limit 3 --serviceName mytest
```

关于获取别名列表的API接口的详细信息，请参见ListAliases。

8.7.7. 自定义域名相关命令

本文介绍FC-API组件中自定义域名相关的命令。

前提条件

您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务
- 创建HTTP函数

创建自定义域名

命令格式：

```
s cli fc-api createCustomDomain --region <regionid> --access <accessname> --props <props> --domainName <
domainName> --protocol <protocoltype> --certConfig <certconfig> --routeConfig <routeconfig>
```

参数说明：

- --region string：指定部署资源的地域。

- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --domainName string：指定自定义域名名称。
- --protocol string：指定自定义域名的协议类型，支持 `HTTP` 和 `HTTP,HTTPS` 协议。
- --certConfig string：配置HTTPS证书的信息。
- --routeConfig string：配置路由表。自定义域名访问时的PATH到Function的映射。

执行示例：

```
s cli fc-api createCustomDomain --region cn-hangzhou --access default --domainName 41055826-188077086902
****.example.com
```

关于创建自定义域名的API接口的详细信息，请参见CreateCustomDomain。

删除自定义域名

命令格式：

```
s cli fc-api deleteCustomDomain --region <regionid> --access <accessname> --props <props> --domainName <
domainName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --domainName string：指定自定义域名名称。

执行示例：

```
s cli fc-api deleteCustomDomain --region cn-hangzhou --domainName 41055826-188077086902****.example.com
```

关于删除自定义域名的API接口的详细信息，请参见DeleteCustomDomain。

更新自定义域名

命令格式：

```
s cli fc-api updateCustomDomain --region <regionid> --access <accessname> --props <props> --domainName <
domainName> --protocol <protocol> --certConfig <certconfig> --routeConfig <routeconfig>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --domainName string：指定自定义域名名称。
- --protocol string：指定自定义域名的协议类型，支持 `HTTP` 和 `HTTP,HTTPS` 协议。
- --certConfig string：配置HTTPS证书的信息。
- --routeConfig string：配置路由表。自定义域名访问时的PATH到Function的映射。

执行示例：

```
s cli fc-api updateCustomDomain --domainName 41055826-188077086902****.example.com --routeConfig {"route
s": [{"path": "/login/*","serviceName": "mytest","functionName": mytest","qualifier":test,"methods":GET
} ]}
```

关于更新自定义域名的API接口的详细信息，请参见UpdateCustomDomain。

获取自定义域名配置信息

命令格式：

```
s cli fc-api getCustomDomain --region <regionid> --access <accessname> --props <props> --domainName <dom
ainName>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --domainName string：指定自定义域名名称。

执行示例：

```
s cli fc-api getCustomDomain --region cn-hangzhou --domainName 41055826-188077086902****.example.com
```

关于获取自定义域名配置信息的API接口的详细信息，请参见GetCustomDomain。

查询自定义域名列表

命令格式：

```
s cli fc-api listCustomDomains --region <regionid> --access <accessname> --props <props> --limit <limit
number> --prefix <return prefix> --nextToken <nextresult> --startkey <startkey>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。
- --prefix string：设置返回资源的名称前缀。
- --startKey string：设定结果从startKey之后（包括startKey）按字母排序的第一个开始返回。

执行示例：

```
s cli fc-api listCustomDomains --region cn-hangzhou --limit 20
```

关于查询自定义域名列表的API接口的详细信息，请参见ListCustomDomains。

8.7.8. 预留配置相关命令

本文介绍FC-API组件中预留配置相关的命令。

前提条件

您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务
- 创建版本
- 创建别名

设置预留配置

命令格式：

```
s cli fc-api putProvisionConfig --region <regionid> --access <accessname> --serviceName <serviceName> --
props <props> --functionName <functionName> --qualifier <qualifier> --target <target parameter> --schedu
ledActions <The configuration of scheduled auto scaling> --targetTrackingPolicies <The configuration of
metric tracking auto scaling>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。
- --target number：指定预留的目标资源个数。
- --scheduledActions string：设置定时伸缩策略。通过定时伸缩策略您可以更加灵活地配置预留的函数实例，在指定时间将预留的函数实例量设定成需要的值，使函数实例量更好地贴合业务的并发量。
- --targetTrackingPolicies string：设置指标伸缩策略。根据函数实例并发利用率的情况每分钟对预留资源进行一次伸缩。

执行示例：

```
s cli fc-api putProvisionConfig --region cn-hangzhou --serviceName mytest --functionName mytest --qualif
ier test --target 4
```

关于设置预留配置的API接口的详细信息，请参见PutProvisionConfig。

获取预留配置信息

命令格式：

```
s cli fc-api getProvisionConfig  --region <regionid> --access <accessname> --props <props> --serviceName
<serviceName> --functionName <functionName> --qualifier <qualifier>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api getProvisionConfig  --region cn-hangzhou --serviceName mytest --functionName mytest  --qual
ifier test
```

关于获取预留配置信息的API接口的详细信息，请参见GetProvisionConfig。

查询预留配置列表

命令格式：

```
s cli fc-api listProvisionConfigs --region <regionid> --access <accessname> --props <props> --limit <lim
it number> --serviceName <serviceName> --qualifier <quelifier> --nextToken <nextresult>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --limit string：设置限定此次返回资源的数量。
- --nextToken string：设置用来返回更多结果的令牌。第一次查询时不需要提供这个参数，后续查询的Token从返回结果中获取。
- --serviceName string：指定服务名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api listProvisionConfigs --region cn-hangzhou --limit 6 --serviceName mytest --qualifier LATEST
```

关于查询预留配置列表的API接口的详细信息，请参见ListProvisionConfigs。

8.7.9. 函数异步调用相关命令

本文介绍FC-API组件中函数异步调用相关的命令。
您已完成以下操作：

- 安装Serverless Devs
- 配置Serverless Devs
- 创建服务
- 创建函数

获取函数异步调用配置信息

命令格式：

```
s cli fc-api getFunctionAsyncConfig --region <regionid> --access <accessname> --props <props> --serviceN
ame <serviceName> --functionName <functionName> --qualifier <version or alias>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api getFunctionAsyncConfig --serviceName mytest --functionName mytest --qualifier test
```

关于获取函数异步调用配置信息的API接口的详细信息，请参见GetFunctionAsyncInvokeConfig。

删除函数异步调用配置

命令格式：

```
s cli fc-api deleteFunctionAsyncConfig --region <regionid> --access <accessname> --props <props> --servi
ceName <serviceName> --functionName <functionName> --qualifier <version or alias>
```

参数说明：

- --region string：指定部署资源的地域。
- --access string：指定使用的密钥别名。
- --props string：配置组件参数。关于props的参数信息，请参见YAML规范。
- --serviceName string：指定服务名称。
- --functionName string：指定函数名称。
- --qualifier string：指定服务的别名或版本。

执行示例：

```
s cli fc-api deleteFunctionAsyncConfig --serviceName mytest --functionName mytest --qualifier test
```

关于删除函数异步调用配置的API接口的详细信息，请参见DeleteFunctionAsyncInvokeConfig。

8.8. 使用Serverless Devs迁移Web框架

本文介绍如何使用Serverless Devs快速迁移各个语言的Web框架。

迁移Java框架





迁移Node.js框架

Custom Runtime

Node.js Runtime









Nuxt.js



Koa.js



Hapi



Midway



Express



Egg.js



Next.js



Nuxt.js



Koa.js



Hapi



Midway

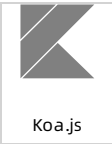


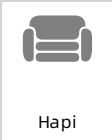
Express

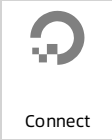

Egg.js

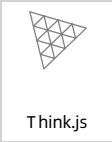

Nest.js


Nuxt.js


Koa.js


Hapi

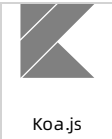

Connect


Think.js


Express


Egg.js


Nuxt.js


Koa.js



Hapi



Connect



Think.js



Nest.js

迁移Python框架



Flask



Tornado



Bottle



Web.py



Django



Pyramid



FastAPI



Flask


Tornado


Bottle


Web.py


Django

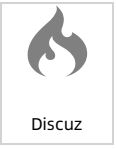

Pyramid


FastAPI

迁移PHP框架


ThinkPHP


Laravel


Discuz


WordPress


Zblog


Ecshop


Metinfo


Whatsns


Typecho


ThinkPHP


Laravel


Discuz


WordPress


Zblog


Ecshop


Metinfo



Whatsns



Typecho

9.常见问题

本文汇总了使用函数计算的常见问题和对应解决方法。

产品简介FAQ

- 我可以访问运行函数的机器吗？
- 非root用户可以免sudo调用Docker吗？
- 我应该什么时候使用函数计算，什么时候使用ECS？

产品计费FAQ

- 如何释放函数计算实例或者停止函数计算服务？
- 性能实例和弹性实例之间可以互相切换吗？
- 我购买的资源套餐包，到期时间是什么时候？
- 是否支持跨地域抵扣？

权限管理FAQ

- 如何让其他用户为我的函数授予访问其他资源的权限？

代码开发FAQ

咨询类FAQ

- 我可以使用什么语言编写函数？
- 函数计算如何保证代码的安全？
- 函数计算只支持Node.js，用C++写的程序怎么运行？
- 函数计算的运行环境中所依赖的包如何自动安装？
- 使用函数Context参数中的AccessKey ID等信息访问其他云资源时，收到The Access Key ID does not exist的报错怎么办？

PHP运行环境FAQ

- 函数计算PHP运行环境支持HTTP触发器吗？
- PHP运行环境支持使用第三方扩展吗？
- PHP运行环境如何加载卸载内置扩展？
- PHP运行环境内置的Tablestore PHP SDK使用有问题怎么办？
- PHP运行环境中Notice或Warning导致某些第三方库（aliyun-openapi-php-sdk）不能使用怎么办？
- 使用PHP运行环境HTTP触发器时，出现Cannot modify header information - headers already sent by (output started at ...怎么办？
- 使用PHP运行环境HTTP触发器时，想更改Session目录怎么办？
- PHP运行环境开发Web时，怎么支持Rewrite？
- 当我需要使用MongoDB等其他PHP非内置扩展怎么办？
- PHP文件中Require_once的使用示例

Custom Runtime FAQ

- Custom Runtime的监听端口一定要和HTTP Server的监听端口一致吗？
- Custom Runtime启动的服务中调用第三方服务时，出现FunctionNotStarted错误怎么办？
- 当我实现的HTTP Server在120s内无法成功启动怎么办？
- 当我使用浏览器或cURL方式访问函数时出现404怎么办？
- 遇502报错且报错信息为Process exited unexpectedly before completing request怎么办？

Custom Container FAQ

- Custom Container的监听端口一定要和HTTP Server的监听端口一致吗？

- Custom Container Runtime启动的服务中调用第三方服务时，出现FunctionNotStarted错误怎么办？
- 当我实现的HTTP Server在120s内无法成功启动怎么办？
- 遇502报错且报错信息为Process exited unexpectedly before completing request怎么办？
- 当我使用浏览器或cURL方式访问函数时出现404怎么办？
- 如未使用阿里云的公网容器镜像，我需要给函数计算的服务角色授予什么权限？

服务管理FAQ

- 什么是VPC？
- 如何用函数访问私有VPC资源？
- 函数可以访问多少个VPC？
- 我在VPC环境里如何访问经典网络中的资源？
- 在VPC中的函数如何访问公网？
- 遇到vSwitch is in unsupported zone的错误怎么办？
- 开启VPC功能对可用性的影响

函数管理FAQ

函数运行FAQ

- 如何监控函数执行？
- 函数可以相互调用吗？
- 可执行文件在本地测试正常，但在函数计算的运行环境中执行时却返回permission denied报错，我该怎么办？
- 如果程序要产生很大一个文件，或者需要一个比较大的空间，在什么地方能申请到？
- 在程序运行过程中，能从哪些接口获取到实际使用的内存？
- 函数如果出现死循环，怎么杀掉函数进程？
- 如何避免循环触发？
- 我的执行环境是否是独立安全的？怎么保障？
- 临时磁盘空间什么时候被释放？
- 当前目录不可写怎么办？
- 我的函数会运行很久，超过函数最大Timeout时间怎么办？
- 我有10个函数，都用到了同一个公共库。我能否只上传一份公共库的代码，让这些函数共享同一份库代码？
- 当我运行Python语言的函数时，遇到报错NoneType object has no attribute split怎么办？
- 函数返回后，执行环境也会随之释放吗？我能复用上一次调用缓存的资源/状态吗？

函数调试FAQ

- 如何快速定位线上问题？
- 为什么我的Node.js函数莫名其妙地超时了？
- 遇到The service or function doesn't belong to you的错误怎么办？
- 遇到SDK.ServerUnreachable：SocketTimeoutException has occurred on a socket read or accept的问题怎么办？
- 为什么通过浏览器访问HTTP Trigger对应的函数会强制下载？
- 如何确定函数内部访问其他服务的网络连通性问题？

函数调用FAQ

- 我的客户端不关心函数执行结果，我不希望我的客户端一直等函数返回怎么办？
- HTTP请求支持异步调用吗？如何获取调用结果？

函数实例FAQ

- 如何弹性扩展函数的执行？
- 函数计算单用户下的按量实例数上限是300，所以我构建的服务每秒最多只能处理300个请求？

访问其他服务FAQ

- 函数计算如何访问RDS？
- 为什么我在函数里无法通过内网Endpoint访问OSS等云服务？
- 在函数中如何访问其他服务？

监控报警FAQ

- 我如何批量下载程序运行过程中产生的日志？
- 函数调用正常，为什么我在监控页面看不到调用次数等指标？

10.联系我们

如果您有任何反馈或疑问，欢迎加入钉钉用户群（钉钉群号：11721331）与函数计算工程师即时沟通。

11.安全白皮书

11.1. 概述

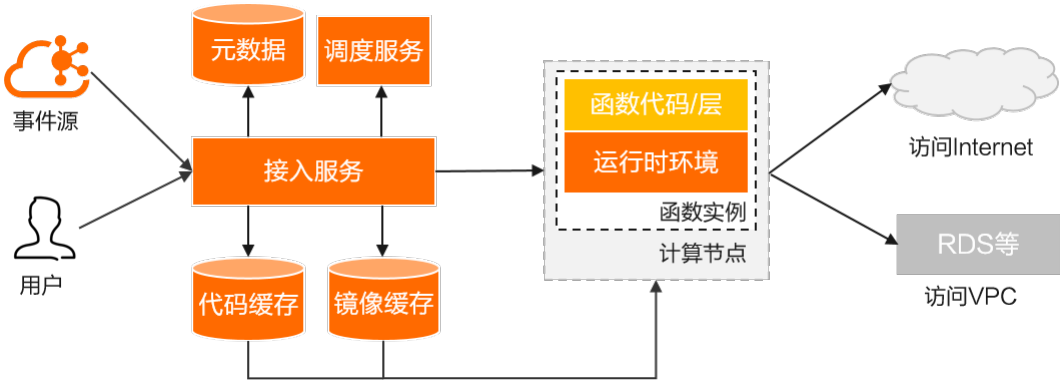
本文从数据安全、用户隐私和用户使用角度介绍函数计算如何打造公共、开放和安全的云计算服务平台。
阿里云函数计算是事件驱动的全托管计算服务。通过函数计算，用户无需管理服务器等基础设施，只需编写代码并上传。函数计算会为用户准备好计算资源，以弹性、可靠的方式运行用户的代码，并提供日志查询、性能监控和报警等功能。

数据安全和用户隐私是阿里云最重要的原则。阿里云致力于打造公共、开放、安全的云计算服务平台。函数计算平台基于阿里云基础服务构建，为用户提供函数接入服务，调度服务，运行时环境等并负责这些服务的安全，而用户需要负责身份凭证以及函数代码、层和配置的安全性，如下图所示。



从用户使用角度，函数计算可分为控制面流程及数据面流程，如下图所示。

- 控制面流程
  包括函数权限控制、代码及配置的增删改查，主要包括函数元数据、代码、层和镜像缓存等安全传输及存储。
- 数据面流程
  函数的调用流程，调用流程包括接入服务、调度服务和计算节点三个模块。
  - 接入服务负责接收用户函数调用请求并发起调用。
  - 调度服务负责计算节点和函数实例的生命周期管理以及调用路由。
  - 计算节点包括多个函数实例，函数实例运行时环境负责执行用户代码。



11.2. 控制面安全性

控制面流程包括函数权限控制、代码及配置的增删改查，主要包括函数元数据、代码、层和镜像缓存等安全传输及存储。
本文介绍函数计算的控制面安全保障类型。

权限控制通过阿里云访问控制（RAM）服务保障安全

- 事件源触发：用户需要为事件源创建触发器，并为触发器授予相应的执行权限才可以触发函数的执行。
- 云服务访问：用户访问其他云服务，比如对象存储OSS、日志服务SLS和表格存储Tablestore时，需要授予相应的访问权限。
- RAM用户（子账号）授权：函数计算借助RAM服务，支持为RAM用户（子账号）赋予不同的函数操作权限。
- 跨账号授权：函数计算借助RAM服务，支持为他人账号赋予不同的函数操作权限。

函数元数据通过传输加密及存储加密保障安全

- 函数计算API及内部通信均使用TLS 1.2及以上协议加密传输内容。
- 函数的元数据通过AES256密文存储，使用时解密后的元数据缓存时间不超过600s。

代码及层缓存通过隔离，受控访问及传输加密保障安全

用户创建或更新函数时，使用OSS同步或API上传等方式，将代码上传至函数计算，函数计算使用隔离的账号缓存代码或层至对象存储OSS。初始化函数实例时，函数计算申请临时下载地址，下载至执行环境，通过虚拟化隔离技术，函数实例只能访问自身的代码及配置的层。

用户可通过调用API、控制台或工具等方式，使用合法的鉴权获取临时下载地址，自行下载代码或层。

代码或层在函数计算内部的传输均使用TLS 1.2及以上协议加密。

镜像缓存通过隔离，受控访问及传输加密保障安全

用户将容器镜像上传至函数计算时，将通过隔离的账号缓存至容器镜像服务ACR中，只有该账号有权限下载镜像。在初始化函数实例时，函数计算使用TLS 1.2及以上加密协议下载容器镜像。

11.3. 数据面安全性

与用户通信时，函数计算使用TLS 1.2及以上协议加密传输调用请求及回包，内部通信使用私有协议防止信息泄露及篡改。本页面分模块阐述数据面安全性保障。

接入服务安全性保障

在数据面内部流程中，接入服务主要负责函数的调用。

作为函数调用的入口，接入服务通过使用负载均衡SLB实现负载均衡及DDoS等网络安全防护。

函数默认只允许公网访问，用户可配置只允许从特定的VPC访问函数，两者之间互斥。

从调用模式角度区分，函数调用可分为同步调用、异步调用和异步任务：

- 同步调用
  一来一回（request-response）调用模式，不会缓存调用请求信息，不会对函数执行错误重试。
- 异步调用
  在接受到调用请求后，会缓存至消息服务MNS，缓存成功后立即返回响应。随后函数计算异步从消息服务MNS中获取请求执行，函数计算保证请求至少执行一次。
  - 不同用户使用的MNS队列至少使用账号级别隔离，对于调用量大的函数，可能使用函数级别隔离。
  - 调用失败时，对于函数执行错误，函数计算会默认重试3次，对于其它错误，如限流，系统错误等，会采用二进制退避方式重试，用户可配置重试次数及最大消息存活时间。
  - 异步调用执行完成时，函数计算支持配置结果回调，用户可以使用结果回调保存调用事件及校验调用结果等。
- 异步任务
  对比异步调用，异步任务提供更丰富的任务控制及可观测能力，用户可以选择终止执行异步任务。具体信息，请参见异步任务。

调度服务安全性保障

在数据面内部流程中，调度服务主要负责计算节点、函数实例的生命周期管理及调用路由。

计算节点

函数计算混合使用ECS神龙裸金属服务器及ECS虚拟机两种机型，可实现用户级动态迁移。

调度服务为每用户默认提供折算总600 CPU逻辑核，1200 GB内存的计算节点，通过池化提供300核或600 GB的内存突发能力，资源池耗尽时，调度服务以不超过360核/分钟的速度为用户扩容，扩容速度超限时会限产生限流错误。用户可通过提交工单申请扩充计算节点上限。

计算节点最大存活时长不超过120小时，当调度服务检测到计算节点异常时，会提前重建计算节点。

函数实例

函数实例分为按需实例及预留实例两种类型，按需实例由函数调用触发动态生产，闲置5分钟时自动释放，预留实例由用户配置初始值及自动扩缩容策略触发生产，闲置时不会释放。

调度服务为每用户默认提供300函数实例突发能力，超过此限制时，以不超过300实例/分钟的速度为用户扩容，扩容速度超限时会产生限流错误，用户可通过提交工单申请扩充函数实例突发上限。

函数实例最大存活时长为不超过36小时，当函数代码或配置发生变化、函数超时、内存超限或客户端主动终止错误时，调度服务会重建容器实例，另外可能由于均衡负载等原因，提前重建函数实例。

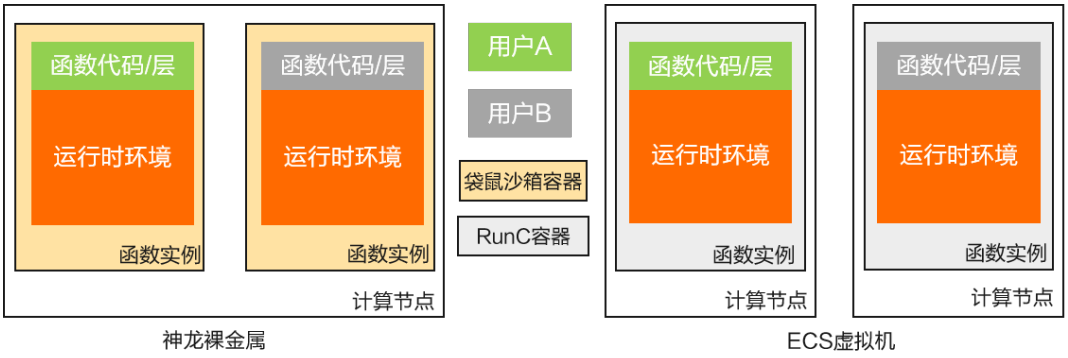
调用路由

调用服务使用 `bin-pack` 算法实现调用路由，同一个函数实例可能被多次函数调用复用，来自同一客户端的调用请求，可能被分发至不同的函数实例执行，用户不可假设函数实例资源，比如全局变量及文件写入或者在不同调用间共享或不共享等。

调度服务会根据用户函数超时配置限制每次函数调用请求占用函数实例的时间，超时后会主动回收函数实例。

计算节点安全性保障

计算节点负责执行用户函数代码，函数计算使用神龙裸金属及ECS虚拟机两种类型的计算节点，如下图所示。本节由外向内分别阐述各层的安全措施。



计算节点层提供阿里云标准的安全防护能力

计算节点提供以下阿里云标准的安全防护能力。具体信息，请参见阿里云安全白皮书。

- 多可用容灾：一个地域的计算节点分布于多个可用区，具备可用区容灾能力。
- 隔离的VPC环境：计算节点位于隔离的VPC环境内，用户不可以直接访问计算节点。
- 云盾安全防护：函数计算节点使用阿里云云盾服务提供漏洞检测，安全防御及对入侵的快速响应与处置能力。
- 漏洞修复或安全升级：函数计算负责计算节点的漏洞修复及安全升级，且升级过程对用户透明。

函数实例提供用户或函数级别的隔离能力

- 虚拟化级别安全隔离
  神龙裸金属计算节点可运行来自不同用户的函数实例，使用阿里云安全沙箱提供函数级别虚拟化及容器隔离，ECS虚拟机只允许运行同用户的函数实例，借助ECS隔离提供用户级别虚拟化隔离，使用Runc等容器技术实现函数级别的容器隔离。
- 函数实例网络访问受限，用户决定网络外访权限
  函数实例配置私有IP地址，用户不可直接访问，且实例间网络不可达，网络隔离使用open vswitch、iptables和routing tables实现。用户可配置4种外部网络访问模式：
  - 仅允许访问公网：函数实例只能访问公网，此模式为默认模式。
  - 仅允许访问VPC：函数实例只能访问配置的VPC网络，比如RDS、NAS和ECS的私有IP地址等。
  - 既允许访问公网，也允许访问VPC：函数实例既能访问公网，也能访问配置的VPC网络。

- 既不允许访问公网，也不允许访问VPC：函数实例完全无访问外部网络的权限。

- 函数实例资源受限
  CPU算力根据内存大小配置成比例分配，但函数实例冷启动期间，允许临时突破限制以加速冷启动，最长持续20s。函数实例默认配置512 MB文件系统容量及1 Gbps网络带宽，如果用户选择性能实例，可最大提供10 GB文件系统容量及5 Gbps网络带宽。
- 函数实例闲时冻结
  当不执行函数请求时，函数实例将被冻结，在下次请求执行前解冻。
- 函数实例允许登录
  通过鉴权的用户可登录名下函数实例，便于线上问题定位。
- 漏洞修复及安全升级
  函数计算负责函数实例沙箱容器的漏洞修复及安全升级，且升级过程对用户透明。

运行时环境辅助用户提升安全能力

- 提供临时身份凭证
  函数计算为用户配置的执行角色申请临时身份凭证，这些凭证将通过环境变量注入运行时环境，且调用时通过输入参数传递至用户代码，用户可通过临时身份凭证访问外部阿里云服务。
- 收集函数执行异常信息
  运行时环境将为用户收集函数执行异常信息及日志，辅助用户识别是否有异常发生。
- 提供生命周期回调提供扩展能力
  运行时环境为用户提供初始化（Initializer）、冻结前（PreFreeze）和停止前（PreStop）等回调，辅助用户按需扩展安全能力。
- 非持久化环境
  运行时环境提供的文件系统及内存，在函数实例释放时随之释放，用户不可使用函数实例的本地文件系统及内存持久化数据，如果用户需要持久化，可以配置阿里云文件存储NAS或持久化保存至对象存储OSS。
- 不可变代码及层（Layer）
  用户对代码目录*/code*及层目录*/opt*的修改，只对本实例生效，不会改写同函数其它实例的代码及库。
- 漏洞修复及安全升级
  运行时环境的漏洞修复及库升级，如果涉及用户兼容性，会通过站内信、短信等方式提前知会用户，对于自定义运行时或自定义容器镜像，运行时的安全保障由用户自行负责。另外对于Java、C#或Golang等编译型语言，函数计算提供的SDK安全漏洞的修复需要用户自行更新依赖并重新编译上传。
- 运行时环境版本支持时间与社区同步
  由于社区对各种运行时环境版本有明确的支持时间，社区不再支持的运行时版本，函数计算会同步给出支持截止时间，会按照禁止新增函数、禁止修改存量函数和禁止函数运行的顺序逐渐停止支持，函数计算不保障停止支持的运行时版本可以一直正常的执行。

11.4. 责任共担

函数计算提供安全机制帮助用户减轻安全保障的负担，用户需要根据自身实际情况，运用好函数计算提供的安全机制来实现业务安全性。

用户身份凭证等鉴权信息不可泄露

如果用户泄露了身份凭证等鉴权信息，业务安全将无从保障。用户可以使用访问控制服务通过RAM用户以最小授权原则授权，结合STS提供临时身份凭证，减少身份凭证等鉴权信息泄露的风险。

函数代码及层安全性保障

- 函数计算为通过鉴权的用户提供临时代码及下载地址，用户需要避免临时下载地址泄露，减少代码或库泄露风险。
- 用户应该避免在代码或层中记录身份凭证等鉴权信息，避免身份凭证信息泄露。
- 用户应关注代码或层的大小，过大可能导致函数实例冷启动过慢或启动失败。
- 由于函数实例可能在不同的调用间复用，前面调用可能导致全局变量、环境变量和文件系统内容等发生修改，用户在代码里不可假设调用在全新的环境执行。
- 由于函数实例不能保障一定在不同调用间复用，用户在代码里不可假设其它调用修改的全局变量，环境变量，文件系统等内容在本次调用依然有效。
- 在异步调用及异步任务场景，函数计算帮助用户自动重试函数执行错误，一个事件保障至少执行一次，但用户不可假设

同事件最多只触发一次调用。

- 当函数执行返回时，函数计算将冻结函数实例，用户不可假设调用返回时所有异步进程、线程、协程等执行完成，也不可假设本次异步写入的日志被刷新。
- 虽然函数计算单个函数实例可以执行多个并发，但并发能力与Runtime语言实现有关系，比如Node.js是单线程模式。Python有GIL全局锁，对计算密集型调用无法并发执行，用户不可假设并发执行数始终符合预期。
- 函数调用有多种资源限制，比如调用请求包和返回包 `headers` 大小不超过4 KB, 环境变量总大小不超过4 KB，请求包 `body` 不超过16 MB等（资源限制可能调整，以官网文档为准），用户不可假设代码在超过资源限制时可以正常执行。
- 函数计算运行时版本支持时间与社区同步，当社区停止支持时，函数计算也会启动停止支持流程，停止支持后不保证函数一直正常运行，用户需要及时弃用已停止支持的运行时版本。
- 用户需要自行保障代码及库的安全性，及时修复漏洞并更新函数。

函数配置安全性保障

函数资源配置

- 用户应合理的设置函数最大执行时长，设置过长在异常时可能产生更多的费用，设置过短可能影响服务可用性。
- 用户应合理设置函数的最大内存配置，设置过高会产生更多的费用，设置过少可能影响服务可用性或执行性能。
- 用户应合理设置单个函数实例的调用并发度，设置过多可能影响服务的可用性或执行性能。
- 用户应合理的设置最大资源配额，避免异常时消耗更多的费用。

函数访问权限配置

- 用户应使用访问控制服务合理的规划不同RAM用户访问函数计算的权限，以降低安全风险。
- 用户应合理的选择函数的访问权限，比如是否限定只能从指定的VPC网络访问。
- 用户应合理的开通触发器的访问权限，以免函数被非预期调用。

函数网络外访配置

用户应根据实际需要合理的选择网络外访配置，比如是否允许访问指定的VPC等，避免敏感信息泄露。

阿里云服务访问配置

- 用户应根据实际需要合理地为函数计算执行角色配置阿里云服务的访问权限，如OSS和OTS等，避免敏感信息泄露。
- 对于用户配置的NAS，当前通过VPC网络传输保障安全性，未设置额外的传输加密。